Adaptive Learning Algorithms for Transferable Visual Recognition

Judy Hoffman

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-139
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-139.html

August 8, 2016

Adaptive Learning Algorithms for Transferable Visual Recognition

by

Judith Hoffman

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Alexei Efros
Professor Jitendra Malik
Professor Bruno Olshausen

Summer 2016

Adaptive Learning Algorithms for Transferable Visual Recognition

Abstract

Adaptive Learning Algorithms for Transferable Visual Recognition

by

Judith Hoffman

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Trevor Darrell, Chair

Understanding visual scenes is a crucial piece in many artificial intelligence applications ranging from autonomous vehicles and household robotic navigation to automatic image captioning for the blind. Reliably extracting high-level semantic information from the visual world in real-time is key to solving these critical tasks safely and correctly. Existing approaches based on specialized recognition models are prohibitively expensive or intractable due to limitations in dataset collection and annotation. By facilitating learned information sharing between recognition models these applications can be solved; multiple tasks can regularize one another, redundant information can be reused, and the learning of novel tasks is both faster and easier.

In this thesis, I present algorithms for transferring learned information between visual data sources and across visual tasks - all with limited human supervision. I will both formally and empirically analyze the adaptation of visual models within the classical domain adaptation setting and extend the use of adaptive algorithms to facilitate information transfer between visual tasks and across image modalities.

Most visual recognition systems learn concepts directly from a large collection of manually annotated images/videos. A model which detects pedestrians requires a human to manually go through thousands or millions of images and indicate all instances of pedestrians. However, this model is susceptible to biases in the labeled data and often fails to generalize to new scenarios a detector trained in Palo Alto may have degraded performance in Rome, or a detector trained in sunny weather may fail in the snow. Rather than require human supervision for each new task or scenario, this work draws on deep learning, transformation learning, and convex-concave optimization to produce novel optimization frameworks which transfer information from the large curated databases to real world scenarios.

To Paul

Contents

Acknowledgments

First and foremost, I want to thank my wonderful PhD advisors, Trevor Darrell and Kate Saenko. You have provided countless hours discussing research, career decisions, and life for which I am truly grateful. Kate, you have been a role model for me, demonstrating a lifestyle I can emulate and a level of career excellence I can aspire to. Trevor, you're support and encouragement has never wavered, giving me an environment to do research which was productive, fun, and allowed continued growth. Thank you both.

I have been extremely fortunate to work with many amazing people during my PhD. Thank you first to the numerous post-docs who have been in our lab and provided advice and interesting research discussions. I specifically want to mention those who I was able to co-author with: Brian Kulis, Erik Rodner, Sergio Guadarrama, Marcus Rohrbach, Ross Girshick and Oscar Beijbom. I would also like to thank Professor Mehryar Mohri for his time and insights working together over the last year on bridging theoretical domain adaptation with practical visual adaptation.

Thank you to my PhD committee members, Trevor Darrell, Alyosha Efros, Jitendra Malik, and Bruno Olshausen for taking the time to provide guidance to me on my dissertation and career choices. I want to offer a special thank you to Jitendra for his encouragement early on in my PhD.

Aside from the many post-docs and professors, I've been surrounded by and worked with so many amazing fellow PhD students. Berkeley computer vision boasts a fantastic group of funny, intelligent, and caring people who have made my PhD experience so memorable, at times entertaining, and always enjoyable. I will miss you all! A special thank you to those who I've worked with closely: Eric Tzeng, Jeff Donahue, Deepak Pathak, Saurabh Gupta, Evan Shelhamer, Ronghang Hu, Jon Long, Ning Zhang, Jian Leong, Yangqing Jia, Kate Rakelly, and Damian Mworca. Thank you also to Georgia Gkioxari, Allie Janoch, and Ning Zhang for your support both early on and throughout this process.

I want to thank my family. To my parents, your support, encouragement, and belief in my success has been a persistent and reliable comfort during the most stressful times. To my brothers, David, Jacob and Ben, thank you for consistently providing levity and perspective for me as I've gone through the whirlwind which is grad school.

Last but not least, to my husband, Paul. You have been my rock and my biggest cheerleader. Your passion for life and intellectual curiosity has inspired me to expand my goals and your endless support has encouraged me to persevere until I achieve them. I can't imagine these last six years without you. My success is our success, thank you for all that you do.

Part I

Introduction and background

Chapter 1

Introduction

1.1 Thesis goals and contributions

The goal of this thesis is to explore the problem of visual domain adaptation. Huge progress has been made in recent years towards solving classic visual recognition problems such as image classification, object detection, and semantic segmentation, however most if not all state-of-the-art solutions require a large number of annotated images and are trained using fully supervised learning techniques. Unfortunately, supervised learning produces a recognition model which is designed to perform well on the exact data it was given at training time, or at least data which is drawn from the same distribution. Often the images or videos that an algorithm will receive at test time differs significantly from the available training data. This means that a recently purchased robot placed into a novel environment won't be able to recognize objects as well as the developers believe based on the robot's performance in it's training environment. Instead of requiring more annotated data from each new environment, the focus of the following research is to propose algorithms for adapting an initial source model for use in a novel target domain, while requiring few or no new annotations.

The thesis proposes solutions for domain adaptation and expands the scope of problem statements for which adaptation solutions may be applied in the following ways:

- Algorithms for adaptation across different visual datasets. We present both a solution for adaption across domains under a fixed representation and when considering adapting both the representation and model space.

- Expanding the notion and understanding of visual domains. We bring to light the heterogenous nature of many visual datasets and propose a solution for discovering latent domains within an unstructured data collection and a separate adaptation solution for data which is temporally structured.

- Combining weak learning and domain adaptation to introduce the concept and algorithms for adaptation across visual recognition tasks. We specifically propose algorithms to adapt a classification model into a detection model.

- Introduce the concept of visual modalities as distinct domains and propose algorithms for effectively transferring information between recognition models for the two visual inputs.

1.2 Organization and previously published work

This thesis is organized as follows. We first briefly review the classical visual recognition problem statements as well as the high level solution methodology of feature computation and/or learning and the most common approaches for recognition models in Chapter 2. Next, we introduce the subject of domain adaptation, describing both the problem statement and the relevant related research in non-visual domain adaptation (Chapter 3).

All previously published algorithms presented in this thesis are work for which I am either lead or co-lead author. I outline the specific algorithms which were previously published below.

Part II focuses on learning transformations across domains using a fixed representation and well as an analysis of adaptation techniques using both shallow and deep representations. This is based on work done with Erik Rodner, Jeff Donahue, Eric Tzeng, Kate Saenko and Trevor Darrell [83, 79, 76].

Part III discusses adaptation simultaneous adaptation of representation and classification models through learning domain invariant image representations. This part builds on work done along with Eric Tzeng, Trevor Darrell, and Kate Saenko [177].

Part IV challenges the notion of a single visual dataset comprising a single visual domain. We present approaches for discovering latent domains, theoretical analysis of multi-source domain adaptation solutions, and describe a continous adaptation approach for use with temporally variable data. This is based on work done with Brian Kulis, Kate Saenko, and Trevor Darrell [82, 77], as well as ongoing work with Mehryar Mohri.

Part V presents two variants on an algorithm for adapting an image classifier into an object detector. This builds on work done in collaboration with Eric Tzeng, Jeff Donahue, Sergio Guadarrama, Trevor Darrell, Kate Saenko, Deepak Pathak, and Ronghang Hu [84, 81] and ongoing work with these collaborators as well as Jonathan Long.

Finally, Part VI of this thesis studies adaptation and transfer of information across visual imaging modalities. Specifically between RGB and depth image representations. We discuss both cross-modal adaptation with limited depth training data and learning to hallucinate depth mid-level information for improving test time RGB detection performance. This contains work done with Saurabh Gupta, Trevor Darrell, Jitendra Malik, Sergio Guadarrama, and Jian Leong [80, 78].

Chapter 2

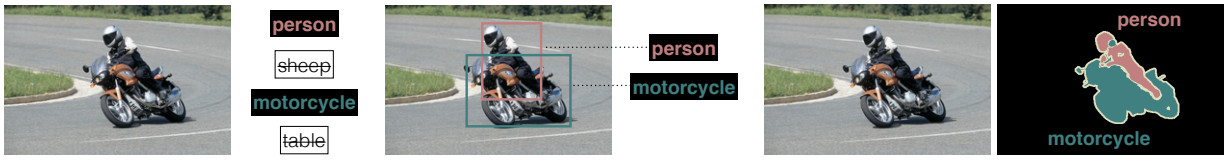
Visual Recognition

2.1 Classic Problem Statements and Datasets

There are three main visual recognition tasks that we focus on in this thesis: classification, detection, and semantic segmentation. All three tasks operate with a fixed label space, being a set of objects that should be recognized within an image. The difference between the task has to do with the amount of spatial feedback an algorithm is expected to produce.

In image classification, the learned model is required to indicate all objects which appear somewhere in a given test image, from the set of known object categories (see Figure 2.1a). Image classification is an integral component of textual/visual search algorithms and automatic photo-organization. Since no spatial information is needed, classification algorithms are most often trained using a collection of training images with corresponding image-level labels indicating the objects which are present in each image. Throughout this document image classification is evaluated in the setting of exactly one known object per evaluation image. We therefore evaluate our approaches using a simple accuracy metric.

Object detection goes one step further than image classification and attempts to roughly localize each recognized object, in the form of a bounding box surrounding the object (see Figure 2.1b). Similarly, the standard supervised data used to train an object detector is bounding box annotations for each known object present in a given image. Since a large component of object detection includes location of objects, most object detection datasets



(a) Classification (b) Detection (c) Semantic Segmentation

Figure 2.1: Illustration of the three types of visual recognition discussed in this thesis.

include more scene-like images each of which contains more than one known object. The extra localization information makes object detection more suited for applications like surveillance and self-driving cars, where it is important to understand the support of a given object. Evaluation of object detection models is done using the mean average precision (mAP) metric. For all experiments, a bounding box proposed by a given algorithm is considered a true positive if it overlaps with the ground truth bounding box annotation by more than 0.5 and no other bounding box proposed by the algorithm with a higher score has already been labeled as a true positive for the overlap object. Average precision is then computed per object category as the area under the precision recall curve as we sweep the score threshold used to produce a predicted bounding box. Finally, the mean average precision is the mean across all object classes of the respective average precisions.

The last classic recognition problem which will be discussed in the text is that of semantic segmentation. In this task the goal is to indicate for each pixel in an image which known object category (or categories) it belongs to (see Figure 2.1c). Usually for these kinds of problems an explicit"background" label is introduced. Note, this task is not the same as instance segmentation where for each pixel both the object class and the particular instance that the pixel belongs to are indicated. Nevertheless, this more spatially specific task proves to be a useful visual model for applications such as robot manipulation where it is important to recognize the surface of any shaped object.

A few of the datasets which will be extensively referred to throughout this document are:

ImageNet A collection of millions of images with tens of thousand object categories. The dataset is a manually curated collection of internet images [35]. The dataset contains classification labels for all images and detection bounding box annotations for around 1000 images per 200 basic level categories.

Office A dataset collected for studying the problem of visual domain adaptation for classification. There are three sub-datasets (domains) each containing objects commonly seen in an office environment [151].

NYUDv2 A dataset of indoor scenes including living rooms, dining rooms and bathrooms. This dataset contains detection and semantic segmentation annotations and each image has both RGB and depth sensory information. [158]

We will discuss each of these datasets as well as all other datasets in more detail in the respective evaluation sections.

The standard approach for producing a recognition model for any one of these tasks is to use a dataset, which contains all the necessary annotations that correspond to the eventual capability of the model, to train parameters of a representation and/or model. In the next section we will briefly cover the representations and models which are referred to through the text.

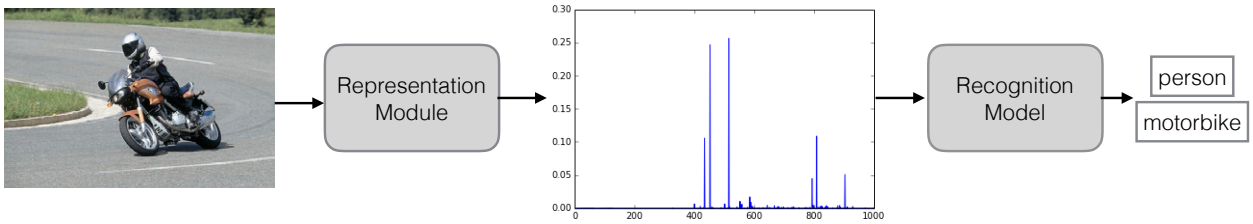


Figure 2.2: Illustration of the classic visual classification architecture using independent feature computation and recognition models.

2.2 Visual Representations and Recognition Models

Most visual recognition systems consist of two components, visual representations and recognition models. The visual representations take as input an image or part of an image and produce a d-dimensional vector which encapsulates the information in the image. The recognition model takes as input the representation vector and indicates which of the known categories are present in the image.

In classic approaches these two components are considered independent modules and are learned or computed separately (see Figure 2.2). Often, the representations, sometimes referred to as features, are hand-designed functions which capture the notion of edge patterns [125, 31, 157]. In this staged approach, the features computed on training images are then used along with known training annotations in order to either learn the parameters of a recognition model in the case of algorithms like support vector machines (SVM) or distances are directly computed in the visual representation space in order to classify an image as in nearest neighbors (kNN).

In recent years, a dominant approach using convolutional neural networks (CNNs) has emerged which combines the representation and model learning into a single objective which is jointly optimized. These architectures take as input all or part of an image and are trained to produce labels for anywhere from the whole image to as specific as every pixel of the image. These methods will be discussed in more detail in Chapter 8.

Chapter 3

Domain Adaptation

3.1 Problem Definition

Standard supervised learning considers being given data, $\mathbf{x}$, and labels, y, drawn from some distribution, $\mathcal{D}$, at training time and fits model parameters so as to minimize some loss between prediction labels, p, and the true known labels, y. A crucial assumption in the supervised learning setup is that new test time data, $\mathbf{x}_{te}$, will be drawn from the same distribution, $\mathcal{D}$, that was seen at training time. Most guarantees about the performance of a model trained in a supervised way are predicated on this assumption.

Domain adaptation instead operates under the explicit assumption of distribution shift between the training and test domain. In particular there is assumed to be a large labeled source domain dataset, $\{\mathbf{x}, y\}$, drawn from the distribution $\mathcal{X}$. However, at test time we assume we will receive data from a distinct target domain with data points, $\mathbf{v}$, drawn from a target distribution, $\mathcal{V}$. The goal of domain adaptation is to learn to adapt the source model for improved performance in the target domain.

This can be accomplished by assuming a small number of labeled examples from all known categories in the target domain, not enough to fully train a robust model in the target domain alone, but enough to learn an adapted source model or a target model regularized by the source data or model. Such approaches are called *supervised domain adaptation*.

In contrast, *unsupervised domain adaptation*, is the task of learning the same adaptation parameters, but without access to any labels from the target domain. Generally, approaches in unsupervised adaptation focus on aligning the marginal empirical distributions through sample re-weighting or non-linear transformations which align the distributions.

For many practical applications, it is unreasonable to assume that the data seen at test time will be drawn from the same distribution as the data collected and annotated for training models. Therefore, adaptation has been explored throughout many areas which use machine learning. In the next section, we'll briefly cover some related work in natural language processing and automatic speech recognition and then in Section 3.3 we will discuss some related work in visual domain adaptation. These sections are meant to introduce the

basic problem statements and general approach paradigms used in all fields. More specific references will be mentioned in the context of each algorithm which is introduced.

3.2 Domain Adaptation for NLP and ASR

Domain adaptation is a critical component of automatic speech recognition (ASR) systems. Here, each new human speaker is considered a distinct domain, where the collection of human speeches which are annotated and used for training a recognition model are considered the multiple source domains while the the particular person who uses the speech recognition system is considered the target domain.

Adaptation has also been extensively studied in the natural language processing community (NLP). Usually, the tasks focus on training on a large language corpora to determine sentiment analysis or document summarization and then the goal is to apply the learned algorithm on related, though distinct, domains. In this case a domain may be reviews of kitchen equipment as a source domain and reviews of books as a target domain. The overall goal of determining sentiment remains fixed, though the language used to describe a kitchen appliance favorably will differ subtly from the language used to describe a book one is happy with.

Some of the earliest adaptation algorithms draw from the economic theory of correction for selection bias [73]. For example, consider a speaker from California who uses the word "like" significantly more frequently than a different English speaker from northern Montana. Thus, a bias will exist in the samples collected in the source domain and will influence the learned model. This type of approach continues to be used for both supervised and unsupervised adaptation in speech recognition and natural language processing [22, 87].

Several methods for adaptation have been developed in the NLP community, e.g., structural correspondence learning was proposed for NLP tasks such as sentiment classification [18]. Daumé III [32] introduced a feature replication method for domain adaptation, which has been applied to both NLP and vision tasks. The basic idea is: given a feature vector $\mathbf{x}$, define the augmented feature vector $\tilde{\mathbf{x}} = (\mathbf{x}; \mathbf{x}; \mathbf{0})$ for data points in the source and $\tilde{\mathbf{v}} = (\mathbf{v}; \mathbf{0}; \mathbf{v})$ for data points in the target. In the linear case, feature replication [32] can be shown to decompose the learned hyperplane parameter into $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}} + \boldsymbol{\theta}'$, where $\hat{\boldsymbol{\theta}}$ is shared by all domains [95].

3.3 Visual Domain Adaptation

The focus of this thesis is on visual domain adaptation. Contrary to domain adaptation in ASR and NLP, visual data suffers the problem of having ill-defined domains. The most standard practice is to assume that each visual dataset collected is considered a single domain and adaptation algorithms are evaluated across datasets. We will present results in this standard setting in Parts II and III. However some of the main contributions of this work

is in expanding the reach of visual adaptation problem statements, both in terms of having a broader definition of visual domains and in terms of recognizing the heterogenous nature of any given dataset and the overlap between multiple datasets when considering the best adaptation algorithms.

Most prior work focuses on the setting of adaptating one visual classifier between two different visual datasets. Many of the first approaches were based on using hand-designed representations with SVM classifiers which would be adapted. For example, [16] proposed a weighted combination of source, target, and transductive SVMs. This is similar to Adaptive SVMs [189, 116], where the target classifier $f^T(\mathbf{x})$ is adapted from the existing, auxiliary classifier $f^A(\mathbf{x})$ via the equation $f^T(\mathbf{x}) = f^A(\mathbf{x}) + \delta f(\mathbf{x})$, where $\delta f(\mathbf{x})$ is the perturbation function. The PMT-SVM method of [7] is related but uses a different regularization term that does not indirectly penalize the margin. Domain transfer SVM [46] attempts to reduce the mismatch in the domain distributions, measured by the maximum mean discrepancy, while also learning a target decision function. A related method [47] utilizes adaptive multiple kernel learning to learn a kernel function based on multiple base kernels.

The disadvantage of methods that only adapt the classifier is their inability to transfer the learned domain shift to novel categories, which is limiting in object recognition scenarios, where the set of available category labels varies among datasets.

So, many approaches were introduced which adapted in a category agnostic way, either by minimizing the distance between the source and target distributions [64, 63, 58], or by learning a category invariant transformation [151, 104, 83, 44].

The most recent approach to visual adaptation involves modifying both the representation and model parameter space of a deep convolutional neural network (CNN). This is commonly done through directly learning a feature space which minimizes the distance between the source and target distributions while simultaneously continuing to correctly classify all available labeled examples [57, 124, 177].

For a more thorough discussion of prior work as it relates to the specific algorithms we present in this thesis, see the relevant chapters surrounding the algorithm introduction.

Part II

Adaptating Across Domains with Fixed Representations

Chapter 4

Introduction and Background

4.1 Problem Setup

In many real-world applications of object recognition, the image distribution used for training (source dataset, or *domain*) is different from the image distribution used for testing (target domain). This distribution shift is typically caused by data collection bias (see Figure 4.1 for three example domains collected for the same set of object categories.) In general, visual domains can differ in a combination of (often unknown) factors, including scene, intra-category variation, object location and pose, viewing angle, resolution, motion blur, scene illumination, background clutter, camera characteristics, etc. Recent studies have demonstrated a significant degradation in the performance of state-of-the-art image classifiers due to domain shift from pose changes [50], a shift from commercial to consumer video [46, 47], and, more generally, training datasets biased by the way in which they were collected [175].

Methods for *adapting* to a target distribution have been proposed, both in and outside of the vision community. Some have focused on learning adapted classifier parameters, typically by minimizing classification error using a small number of (category) labels in the target domain [16, 95, 189]. Others use existing classifiers but learn a transformation between the *features* in the various domains, either by utilizing unlabeled but corresponding points across domains, such as a scene captured simultaneously from multiple views [50], or by somehow aligning the domain distributions [64, 63].

In this paper, we introduce a novel domain adaptation technique that improves on and combines the above approaches. We first propose a novel method of learning a domain-invariant feature space, and later extend this formulation to simultaneously adjust the decision boundary *in the new space*, using all available labeled data.

The key idea behind our feature adaptation method is to learn a *regularized transformation* that maps feature points from one domain to another using cross-domain constraints. The constraints are formed by requiring that the transformation maps points from the same category (but different domain) near each other. Its advantages over previous feature adaptation methods are that 1) it can learn from category labels, and not just from instance-level

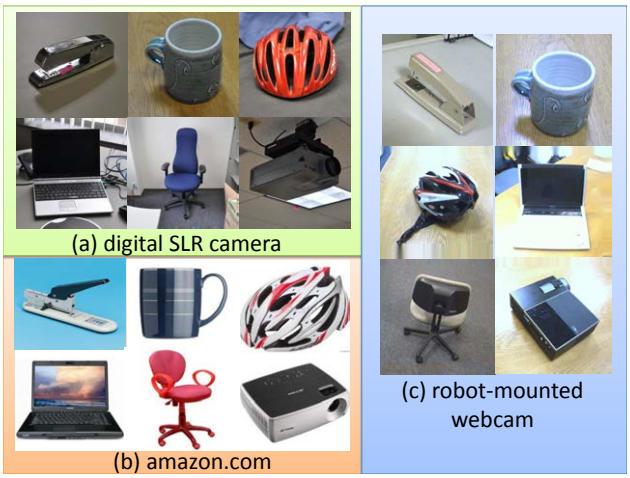


Figure 4.1: We address the problem of transferring object category models between visual domains, such as (a) high-resolution DSLR photographs of objects taken by a human, (b) images downloaded from amazon.com, and (c) images captured by a robot-mounted webcam. Each domain is characterized by a distinct feature distribution caused by, e.g., background clutter in (a,c) vs. uniform backgrounds in (b), or fine-grained detail in (a,b) vs. lack thereof in (c) (as on the laptop keyboard).

constraints, 2) it can adapt models between heterogeneous spaces, including those with different dimensions, via an *asymmetric transform*, and 3) the learned transformation is category independent and thus transferrable to unlabeled categories. This method, which we call the **Asymmetric Regularized Cross-domain Transform (arc-t)**, is independent of the classifier and allows us to encode domain invariance into the feature representation of a broad range of classification methods, from k-NN to SVM, as well as clustering methods.

Forcing all intra-class points to be similar can be inefficient when the end goal is to learn a decision boundary. We extend the above to simultaneously learn the transformation of features and the *classifier parameters* themselves, using the same classification loss to jointly optimize both. This method, referred to as **Maximum Margin Domain Transform (mmdt)**, provides a way to adapt max-margin classifiers in a multi-class manner, by learning a shared component of the domain shift as captured by the feature transformation.

Because it operates over the input features, **mmdt** can generalize the learned shift in a way that parameter-based methods cannot. On the other hand, it overcomes the limitations of the **arc-t** method as applied to classification: by optimizing the classification loss directly in the transform learning framework, it can achieve higher accuracy; furthermore, its use of efficient hyperplane constraints significantly reduces the training time of the algorithm, and learning a transformation directly from target to source allows efficient optimization in linear space.

The article builds on several conference publications. The transform learning methods for domain adaptation have been presented in [151] (for symmetric metrics) and in [104] (asym-

metric **arc-t**). The max-margin formulation was introduced in [83]. This work presents a unified framework for all three methods, and further insights into their underlying connections. In addition, we present a comprehensive comparison of the methods to each other, as well as to recent visual domain adaptation approaches.

4.2 Prior Work

Domain adaptation is a fundamental problem in machine learning and in related fields. It has attracted a lot of attention in the natural language community [18, 32, 11, 94] and in computer vision [16, 115, 91, 63].

The problem statement of domain adaptation is related to multi-task learning, but differs from it in the following way: in domain adaptation problems, the distribution $p(\boldsymbol{x})$ over the features varies across tasks (domains), while the output labels y remain the same; in multi-task learning or knowledge transfer, $p(\boldsymbol{x})$ stays the same across tasks (single domain), while the output labels vary (see [93] for more details). In this article, we address *multi-task learning across domains*; i.e., both $p(\boldsymbol{x})$ and the output labels y can change between domains.

In the following, we briefly review domain adaptation methods that either focus on computer vision applications or that are related to our approach. We present a detailed comparison to several of these methods in Section 5.5. A comprehensive overview of multi-task learning and domain adaptation is given in [93].

Classifier adaptation Several classifier-centric approaches have been presented for domain adaptation, most based on the SVM. For example, [16] propose a weighted combination of source, target, and transductive SVMs. One of the prominent approaches is given by Daumé III [32], who introduces a feature replication method for domain adaptation (as described in Section 3.3). Daumé III also gives an overview of the relevant baselines, which we employ in this work. This is similar to Adaptive SVMs [189, 116], where the target classifier $f^T(\boldsymbol{x})$ is adapted from the existing, auxiliary classifier $f^A(\boldsymbol{x})$ via the equation $f^T(\boldsymbol{x}) = f^A(\boldsymbol{x}) + \delta f(\boldsymbol{x})$, where $\delta f(\boldsymbol{x})$ is the perturbation function. The PMT-SVM method of [7] is related but uses a different regularization term that does not indirectly penalize the margin. Domain transfer SVM [46] attempts to reduce the mismatch in the domain distributions, measured by the maximum mean discrepancy, while also learning a target decision function. A related method [47] utilizes adaptive multiple kernel learning to learn a kernel function based on multiple base kernels.

The disadvantage of methods that only adapt the classifier is their inability to transfer the learned domain shift to novel categories, which is limiting in object recognition scenarios, where the set of available category labels varies among datasets.

Multi-view learning Multi-view learning [154, 143, 51, 37] addresses the scenario where multiple sets of observations are available per labeled example, resulting in multiple *views*

of the data. The views could come from different modalities or, in vision, different 3D pose of the same object instance. For visual domain adaptation, such methods have been applied in cases where multiple observations of the same instance are available [50, 30, 115]. For example, [97] use multi-view learning on multiple views of the same face to perform face recognition across pose variation. In contrast to classifier adaptation described above, such methods attempt to learn a perturbation over the feature space, rather than a class-specific perturbation of the model parameters, typically in the form of a transformation matrix or modified kernel. In particular, [50] as well as [115] translate features between camera views to transfer activity models, while [30] translated between text and image domains.

Our method can handle multiple views, i.e. data with *instance constraints*, when available; however, unlike multi-view learning, it can also handle the case of multiple object category datasets that have no instances in common, and only share the same category labels.

Hybrid supervised methods Instead of choosing either feature transformation or classifier adaptation, it is possible to combine the two approaches, as we do in this paper with **mmdt**. The approach most closely related to ours is the recent Heterogeneous Feature Augmentation (HFA) method [44], which learns a feature transformation into a common latent space, as well as the classifier parameters. However, in contrast to **mmdt**, **hfa** is formulated to solve a binary problem, so a new feature transformation must be learned for each category. Therefore, unlike **mmdt**, **hfa** cannot learn a representation that generalizes to novel target categories. Furthermore, our method has better computational complexity.

Semi- and Un-supervised methods While we do not discuss in detail it here, in [41] we presented a semi-supervised extension of our model, adding constraints between unlabeled target points to the labeled constraints. For example, we placed smoothness constraints on examples that lay on a consistent motion path and could thus be hypothesized to have *the same*, albeit unknown, label. Domain adaptation in a purely unsupervised setting (no labeled target domain examples) has been considered by [64] and [63]. The main idea of both works is to build subspaces for the source as well as the target domain and to consider the path between them on the corresponding manifold. A new feature representation is calculated by concatenating intermediate subspaces on the path. Whereas, [64] samples a finite set of intermediate subspaces, the Geodesic Flow Kernel (**gfk**) of [63] shows how to use all subspaces on the geodesic path by using a kernel trick. This yields a symmetric kernel for source and target points that can be used for example for nearest neighbor classification. More recently, [25] extended this framework to handle image features learned using deep convolutional neural networks.

Chapter 5

Category Invariant Feature Transformations

5.1 Framework Introduction

In this chapter we present a cohesive framework for learning a single transformation matrix $\boldsymbol{W}$ which maps examples between the source and target domains. The objective for the transformation is to diminish domain-induced differences so that examples can be compared directly. This mapping step can then be followed by standard distance-based learning.

We denote the source domain as $\mathcal{X}$ and the target domain as $\mathcal{V}$. Similarly, we denote the source data points as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_\mathcal{X}}] \in \mathbb{R}^{d_\mathcal{X} \times n_\mathcal{X}}$ with labels $\boldsymbol{y} = [y_1, \ldots, y_{n_\mathcal{X}}]$ and the target data points as $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{n_\mathcal{V}}] \in \mathbb{R}^{d_\mathcal{V} \times n_\mathcal{V}}$ with corresponding labels $\boldsymbol{g} = [g_1, \ldots, g_{n_\mathcal{V}}]$. The target domain is assumed to have significantly fewer labeled examples than the source, i.e. $n_\mathcal{V} \ll n_\mathcal{X}$, and there may even be some categories for which the target domain has *no* labeled examples. We use whichever categories have labeled target examples to learn a transformation that is generalizable across categories and so can be applied to *all* categories at test time. Note that our algorithm allows the source and target feature spaces to have different dimensions ($d_\mathcal{X} \neq d_\mathcal{V}$).

To learn such a transformation we will define a matrix regularizer, $r(\boldsymbol{W})$ and a loss, $\mathcal{L}(\boldsymbol{W}, \mathbf{X}, \mathbf{V}, \boldsymbol{y}, \boldsymbol{g})$, which are computed as some function of the category labeled source and target data[1]. With these two terms defined we solve the following general optimization problem:

$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \quad r(\boldsymbol{W}) + \lambda \cdot \mathcal{L}(\boldsymbol{W}, \mathbf{X}, \mathbf{V}, \boldsymbol{y}, \boldsymbol{g}) \tag{5.1}$$

[1]Note that in general we could equally optimize a second loss function between the source and target data which considers *instance* level constraints. However, to distinguish ourselves from prior work which focused on learning a metric requiring instance constraints, we present our algorithms assuming only category level information to demonstrate the effectiveness of using only this coarser level of supervision.

5.2 Category Invariant Feature Transformations through Similarity Constraints

Learning a transformation can be also viewed as learning a similarity function between source and target points, $\text{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v}) = \mathbf{x}^T \boldsymbol{W} \mathbf{v}$. This perspective was used by **arc-t** [151, 104], which extended metric learning techniques [33] towards a domain adaptation scenario. Intuitively, a desirable property of this similarity function is that it should have a high value when the source and target points are of the same category and a low value when the source and target points are of different categories.

This intuitive goal can be formulated by constructing a constraint for each pair $(\{\mathbf{x}, y\}, \{\mathbf{v}, g\})$ of labeled source and target points:

$$c(\boldsymbol{W}, \mathbf{x}, \mathbf{v}, y, g) \quad := \quad \begin{cases} \text{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v}) > u & y = g \\ \text{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v}) < l & y \neq g \end{cases} , \tag{5.2}$$

for some constants, $u, l \in \mathbb{R}$.

If optimized, the constraints specified in Equation (5.2) guarantee that source and target points with the same label have high similarity and that source and target points with different labels have low similarity.

In general, we do not need each pairwise constraint to be satisfied to learn a good similarity function, therefore, we optimize soft constraints in the form of the following loss function:

$$\ell(\boldsymbol{W}, \mathbf{x}, \mathbf{v}, y, g) \quad = \quad \begin{cases} \max(0, \text{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v}) - u) \\ \qquad \text{if } y = g \\ \max(0, l - \text{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v})) \\ \qquad \text{if } y \neq g \end{cases} \tag{5.3}$$

Finally, we define a loss for all labeled data points as the squared sum over each pairwise loss:

$$\mathcal{L}(\boldsymbol{W}, \mathbf{X}, \mathbf{V}, \boldsymbol{y}, \boldsymbol{g}) = \sum_{i,j} [\ell(\boldsymbol{W}, \mathbf{x}_i, \mathbf{v}_j, y_i, g_j)]^2 \ . \tag{5.4}$$

Using this loss function in the general framework of Equation (5.1), we seek to solve the following optimization problem:

$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \ r(\boldsymbol{W}) + \sum_{i,j} [\ell(\boldsymbol{W}, \mathbf{x}_i, \mathbf{v}_j, y_i, g_j)]^2 \ . \tag{5.5}$$

Constraints thus far have been defined for category level correspondences, however, if additional paired instance correspondence is available for some data, this auxiliary information could be incorporated using the same similarity constraint technique. In [41], we also showed that constraints between labeled and unlabeled target points can help learning in a semi-supervised fashion.

An important second part of the objective function as defined in Section 5.1 is the regularizer of the transformation matrix. This term contains our prior knowledge about the transformation and has to be chosen carefully. We present two types of very flexible regularization terms in the following sections.

LogDet Regularizer for Symmetric Transforms

We will begin by considering the log determinant (LogDet) regularizer:

$$r(\boldsymbol{W}) = \mathrm{tr}(\boldsymbol{W}) - \log\det(\boldsymbol{W}) \tag{5.6}$$

for positive definite matrices $\boldsymbol{W}$.

Using the LogDet regularizer causes the formulation in Equation (5.5) to become equivalent to that of Information-theoretic Metric Learning (ITML), which indirectly learns a transformation matrix $\boldsymbol{W}$ corresponding to a linear transformation between $\mathcal{X}$ and $\mathcal{V}$ by optimizing the pairwise loss functions given in Equation (5.5).

With this regularization term, the optimization function is kernelizable and a final nonlinear transformation can be learned to map between the source and target points.

While this model is intuitively appealing for domain adaptation, it requires a key simplifying assumption that the source and target data have the same dimension; i.e., $d_{\mathcal{X}} = d_{\mathcal{V}}$. This follows from the fact that the matrix trace and determinant are only defined for square matrices $\boldsymbol{W}$. An even stronger restriction on $\boldsymbol{W}$ made by the LogDet regularizer is that it is only defined over symmetric positive definite matrices. Implicitly, if $\boldsymbol{W}$ is positive definite then $\boldsymbol{W}$ can be decomposed into the product of two identical matrices - $\boldsymbol{W} = \boldsymbol{R}^T\boldsymbol{R}$. Therefore, the similarity function learned can be decomposed into:

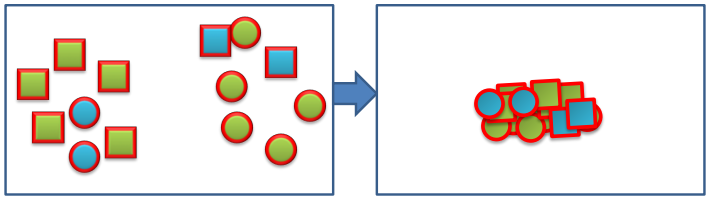
$$\mathrm{sim}(\boldsymbol{W}, \mathbf{x}, \mathbf{v}) \;=\; \mathbf{x}^T\boldsymbol{W}\mathbf{v} = (\boldsymbol{R}\mathbf{x})^T(\boldsymbol{R}\mathbf{v}) \tag{5.7}$$

The observation here is that a symmetric positive definite matrix corresponds to applying the same transformation to the source and target domains.

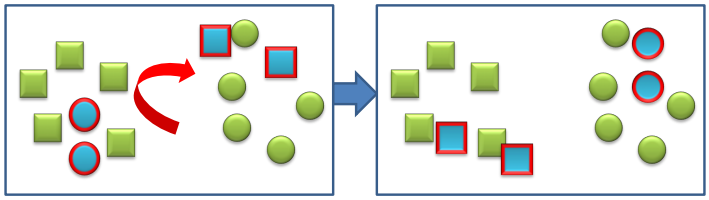
Using the LogDet regularizer limits the applicability of domain adaptation due to the restricted class of possible transformation matrices $\boldsymbol{W}$. Consider the scenario in Figure 5.1, where there is no symmetric transformation that can transform between the source and target domains. In the next section, we will mitigate this limitation.

Frobenius Regularizer for Asymmetric Transforms

In order to avoid the restrictions of the symmetric transformation model for adaptation, we seek an alternative regularizer that allows the model to be applied to domains of differing dimensionalities but that still retains the benefits of kernelization. We choose the Frobenius norm regularizer, which is defined for general matrices $\boldsymbol{W}$ in asymmetric transformations (Figure 5.1). We call this problem the **Asymmetric Regularized Cross-domain transformation** problem with similarity and dissimilarity constraints, or **arc-t** for short, in the rest of the paper.



(a) A symmetric transformation – the same rotation and scaling applied to both domains (green and blue) – cannot separate classes (circles and squares)



(b) An asymmetric transformation – a rotation applied only to blue domain – successfully compensates for domain shift

Figure 5.1: A conceptual illustration of how an asymmetric domain transformation matrix corresponding to a linear transformation can be more flexible than a symmetric one.

Using the loss function defined in Equation (5.5), our new optimization objective is given as follows:

$$\min_{\boldsymbol{W}} \quad \frac{1}{2}\|\boldsymbol{W}\|_F^2 + \lambda \sum_{i,j} [\ell(\boldsymbol{W}, \mathbf{x}_i, \mathbf{v}_j, y_i, g_j)]^2 \tag{5.8}$$

There are two main limitations of the transformation learning problem (5.8) presented above. First, it is limited to linear transformation matrices, which may not be sufficient for some adaptation tasks. Second, the size of $\boldsymbol{W}$ grows as $d_{\mathcal{X}} \cdot d_{\mathcal{V}}$, which may be prohibitively large for some problems. To overcome both these shortcomings, a kernelization result was presented by Kulis *et al.* [104].

Whether using the linear or kernelized version of the algorithm, the general idea of using pairwise constraints to learn $\boldsymbol{W}$ limits the ability of this learning algorithm to scale with the number of labeled points in the source and target, since the number of constraints generated is $n_{\mathcal{X}} \cdot n_{\mathcal{V}}$. Additionally, $\boldsymbol{W}$ is learned so as to place source and target points close if they are of the same category and far if they are from different categories. While this is an intuitive notion, it fails to directly optimize the overall objective of correctly classifying target points. In the next section, we describe an alternate approach which overcomes these limitation by jointly learning $\boldsymbol{W}$ and classifier parameters.

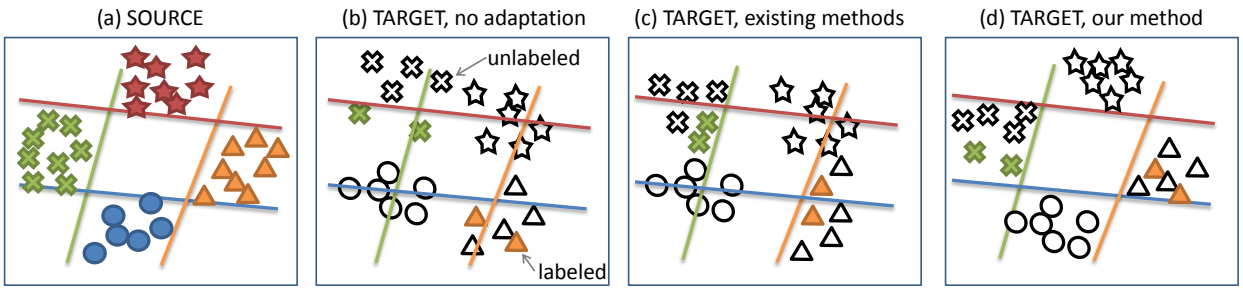


Figure 5.2: (a) Linear classifiers (shown as decision boundaries) learned for a four-class problem on a fully labeled source domain. (b) Problem: classifiers learned on the source domain do not fit the target domain points shown here due to a change in feature distribution. (c) Existing SVM-based methods only adapt the features of classes with labels (crosses and triangles). (d) Our method adapts all points, including those from classes without labels, by transforming all target features to a new domain-invariant representation.

Optimization

In this section, we briefly describe optimization techniques that can be used to solve the objectives described.

When optimizing the objective with a LogDet norm regularizer, we use the standard ITML method as mentioned in Section 5.2. To optimize the objective with the Frobenius norm regularizer, we use one of two methods. First, note that the problem can actually be reformulated as a quadratic programming (QP) problem, after which any standard QP solver can be used to optimize our objective. However, if the size of $\boldsymbol{W}$ is too large, this is impractical. A separate approach is to use the Bregman divergence method [33]. Both techniques yield similar performance, but have varying convergence rates.

5.3 Category Invariant Feature Transformations through Optimizing Classification Objective

In this section, we present a different loss function that can be used within the transform-based domain adaptation framework defined in Equation (5.1). The goal now is to directly optimize a classification objective for the target points, while simultaneously presenting a learning algorithm that is more scalable with the number of labeled source and target points (Figure 5.2).

For our algorithm, we consider linear hyperplane classifiers. For example, assume that a one-versus-all linear SVM classifier has been trained on the labeled source data over all K categories. Let $\boldsymbol{\theta}_k$ denote the normal to the hyperplane associated with the k'th binary SVM problem. Similarly, let b_k be the offset to the hyperplane associated with the k'th binary SVM problem. Finally, let $\tilde{\boldsymbol{\theta}}_k^T = \begin{bmatrix}\boldsymbol{\theta}_k^T & b_k\end{bmatrix}$ be the full affine hyperplane representation.

Intuitively, we seek to learn a transformation matrix $\boldsymbol{W}$ such that once $\boldsymbol{W}$ is applied to the target points, they will be classified accurately by the source SVM. We consider learning an affine linear transformation matrix, which can be easily done using homogeneous coordinates for our data points: $\tilde{\mathbf{v}}^T = \begin{bmatrix}\mathbf{v}^T & 1\end{bmatrix}$.

The key idea is now to use constraints based on linear classifiers instead of single instances. In particular, we require that transformed target points are correctly classified in the source domain:

$$c(\boldsymbol{W}, \tilde{\boldsymbol{\theta}}_k, \tilde{\mathbf{v}}, g) \quad := \quad \mathbb{I}(g = k) \left(\tilde{\boldsymbol{\theta}}_k^T \boldsymbol{W} \tilde{\mathbf{v}} \right) \geq 1 \ , \tag{5.9}$$

where $\mathbb{I}$ is the signed indicator function, with $\mathbb{I}(z) = 1$ when z is true and $\mathbb{I}(z) = -1$ in the other case. If Equation (5.9) is fulfilled, all transformed target points would be correctly classified by the source linear classifier. However, this is only possible for separable cases, so instead we optimize the soft constraints in form of the hinge loss:

$$\ell(\boldsymbol{W}, \tilde{\boldsymbol{\theta}}_k, \mathbf{v}_i, g_i) \quad = \quad \max(0, 1 - \mathbb{I}(g_i = k) \cdot \tilde{\boldsymbol{\theta}}_k^T \boldsymbol{W} \tilde{\mathbf{v}}_i) \tag{5.10}$$

Similarly, if we use $\boldsymbol{\Theta} = \begin{bmatrix}\tilde{\boldsymbol{\theta}}_1 & \dots & \tilde{\boldsymbol{\theta}}_K\end{bmatrix}$ to denote all hyperplane parameters of the one-versus-all classifier, the loss over all target points and all categories is given as:

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{\Theta}, \mathbf{V}, \boldsymbol{g}) \quad = \quad \sum_{k,i} \ell(\boldsymbol{W}, \tilde{\boldsymbol{\theta}}_k, \mathbf{v}_i, g_i) \tag{5.11}$$

Because the target points are transformed into the source domain space with $\boldsymbol{W}$, we simply define the source data term in our loss function as standard SVM hinge loss summed over all categories:

$$\mathcal{L}(\boldsymbol{\Theta}, \mathbf{X}, \boldsymbol{y}) \quad = \quad \sum_{k,i} \max(0, 1 - \mathbb{I}(y_i = k) \cdot \tilde{\boldsymbol{\theta}}_k^T \mathbf{x}_i) \tag{5.12}$$

Once the transformation matrix $\boldsymbol{W}$ has been learned, we can also use it to transform linear classifiers $\tilde{\boldsymbol{\theta}}_k$ to the target domain that had been learned with source data only. This is a huge advantage of modelling the domain shift as being category-invariant, because we only need a few categories present in both target and source training data and are able to transfer all available category models in the source domain to the target domain. For regularization of $\boldsymbol{W}$, we use the Frobenius norm regularizer for this optimization problem. Optimizing this objective in Equation (5.1) using the loss in Equation (5.10) and the Frobenius norm regularizer leads to a category invariant and asymmetric transformation matrix, which considers classifier constraints in the source domain. Additionally, the learning algorithm no longer has a linear dependency on the number of source training examples and instead scales with the number of categories and the number of labeled target points, $K \cdot n_{\mathcal{V}}$.

5.4 Jointly Optimizing Classifier and Transformation

Our goal in this section is to jointly learn 1) affine hyperplanes that separate the categories in the common domain consisting of the source domain and target points projected to the source and 2) the new feature representation of the target domain determined by the transformation matrix $\boldsymbol{W}$ mapping points from the target domain into the source domain.

The algorithm and the change of constraints presented in the previous section is especially useful when linear classifiers are already learned in the source domain. However, we can also formulate a joint learning problem for the transformation matrix and the classifier parameters; i.e., the hyperplane parameters and thus the decision boundary are also affected by the additional training data provided from the target domain.

The transformation matrix should have the property that it projects the target points onto the correct side of each source hyperplane and the joint optimization also maximizes the margin between two classes. Therefore, we refer to this method as **Maximum Margin Domain Transform**, or **mmdt**.

The joint optimization problem can be formulated by adding a regularizer on $\boldsymbol{\Theta}$.

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{\Theta}} \quad & \frac{1}{2}\|\boldsymbol{W}\|_F^2 + \frac{1}{2}\|\boldsymbol{\Theta}\|_F^2 + \lambda\,\mathcal{L}(\boldsymbol{W},\boldsymbol{\Theta},\mathbf{V},\boldsymbol{g}) \\ & +\lambda_{\mathcal{X}}\,\mathcal{L}(\boldsymbol{\Theta},\mathbf{X},\boldsymbol{y}) \end{aligned} \tag{5.13}$$

In contrast to the previous optimization problems, the problem in Equation (5.13) is no longer convex. For this reason, we perform coordinate gradient descent by alternating between optimizing with respect to $\boldsymbol{W}$ and $\boldsymbol{\Theta}$:

1. Initialize $\boldsymbol{\Theta}^0$ using a 1-vs-all SVM trained on the source data only.
2. Learn $\boldsymbol{W}^t$ assuming fixed $\boldsymbol{\Theta}^t$.
3. Learn $\boldsymbol{\Theta}^{t+1}$ assuming fixed $\boldsymbol{W}^t$.
4. Iterate between (2)-(3), until convergence.

Note that step (2) is equivalent to solving the optimization problem presented in Section 5.3. Additionally, note that step (3) is equivalent to solving a multi-category SVM problem defined over source and transformed target data points. This can again be solved using K 1-vs-all binary SVM classifiers.

An important property of the alternating optimization is that we can indeed prove convergence by exploiting the convexity of both sub-problems.

Lemma 1. *Steps (2) and (3) will never increase the complete joint objective function.*

Proof. Let $J(\boldsymbol{W},\boldsymbol{\Theta})$ denote the value of the joint objective function.

Claim 1: $J(\boldsymbol{W}^t, \boldsymbol{\Theta}^t) \geq J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^t)$

$$J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^t) \;\; = \;\; \min_{\boldsymbol{W}} J(\boldsymbol{W}, \boldsymbol{\Theta}^t) \leq J(\boldsymbol{W}^t, \boldsymbol{\Theta}^t)$$

Claim 2: $J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^t) \geq J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^{t+1})$

$$J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^{t+1}) = \min_{\boldsymbol{\Theta}} J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}) \leq J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^t)$$

The key here is that steps (2) and (3) of our algorithm are convex optimization problems and so we know that each objective will never increase as the new variable values are learned. □

Theorem 2. *The joint objective function for Eq.* (5.13) *will converge.*

Proof. Using Lemma (1), we can directly show that the joint objective function will not increase from one iteration to the next:

$$J(\boldsymbol{W}^t, \boldsymbol{\Theta}^t) \;\; \geq \;\; J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^t) \geq J(\boldsymbol{W}^{t+1}, \boldsymbol{\Theta}^{t+1})$$

Additionally, since the joint objective is lower bounded by zero, this proves that the joint objective will converge for a sufficiently small step size if optimizing using gradient descent. □ □

It is important to note that since both steps of our iterative algorithm can be solved using standard QP solvers, the algorithm can be easily implemented. Furthermore, we also developed a fast optimization technique based on dual coordinate descent and exploiting an implicit rank contraint of $\boldsymbol{W}$ in [148]. The method allows using the MMDT algorithm even in large-scale scenarios with tens of thousands of examples and high-dimensional features, because not all of the entries of $\boldsymbol{W}$ have to be optimized.

5.5 Analysis

We now analyze and compare the proposed algorithms against each other and the previous feature transform methods **hfa** [44] and **gfk** [63]. Comparisons are summarized in Table 5.1.

The **arc-t** formulation, of Section 5.2, has two distinct limitations. First, it must solve $n_{\mathcal{X}} \cdot n_{\mathcal{V}}$ constraints, whereas **mmdt**, of Section 5.4, only needs to solve $K \cdot n_{\mathcal{V}}$ constraints, for a K category problem. In general, **mmdt** scales to much larger source domains than **arc-t.** The second benefit of the **mmdt** learning approach is that the transformation matrix learned using the max-margin constraints is learned jointly with the classifier, and explicitly seeks to optimize the final SVM classifier objective. While **arc-t**'s similarity-based constraints seek to map points of the same category arbitrarily close to one another, followed by a separate classifier learning step, **mmdt** seeks simply to project the target points onto the correct side of the learned hyperplane, leading to better classification performance.

	hfa [44]	**gfk** [63]	**symm** [151]	**arc-t** [104]	**mmdt** [83]
multi-class		✓	✓	✓	✓
large datasets		✓			✓
heterogeneous features	✓			✓	✓
optimize max-margin objective	✓				✓

Table 5.1: Unlike previous methods (**hfa** by [44] and **gfk** by [63]), our final approach using max-margin constraints and Frobenius norm regularizer is able to simultaneously learn multi-category representations that can transfer to novel classes, scale to large training datasets, and handle different feature dimensionalities.

The **hfa** formulation [44] also takes advantage of the max-margin framework to directly optimize the classification objective while learning transformation matrices. **hfa** learns the classifier and transformations to a common latent feature representation between the source and target. However, **hfa** is formulated to solve a binary problem so a new feature transformation must be learned for each category. Therefore, unlike **mmdt**, **hfa** cannot learn a representation that generalizes to novel target categories. Additionally, due to the difficulty of defining the dimension of the latent feature representation directly, the authors optimize with respect to a larger combined transformation matrix and a relaxed constraint. This transformation matrix becomes too large when the feature dimensions in source and target are large, so the **hfa** problem must usually be solved in kernel space. This can make the method slow and cause it to scale poorly with the number of training examples. In contrast, **mmdt** can be efficiently solved in linear feature space which makes it fast and potentially more scalable.

Finally, **gfk** [63] formulates a kernelized representation of the data that is equivalent to computing the dot product in infinitely many subspaces along the geodesic flow between the source and target domain subspaces. The kernel is defined to be symmetric, so it cannot handle source and target domains of different initial dimension. Additionally, **gfk** does not directly optimize a classification objective. In contrast, **mmdt** can handle source and target domains of different feature dimensions via an asymmetric $\boldsymbol{W}$, as well as directly optimizing the classification objective.

5.6 Datasets

We begin by introducing the data on which we will evaluate our algorithms.

Office database

In most of our experiments, we consider the *Office* database first introduced by [151], which has become the de facto standard for benchmarking visual domain adaptation methods.

This database allows researchers to study, evaluate and compare solutions to the domain shift problem by establishing a multiple-domain labeled dataset and benchmark. In addition to the domain shift aspects, this database also proposes a challenging office environment category learning task which reflects the difficulty of real-world indoor robotic object recognition. It contains images originating from the following three domains:

Images from the web: The first domain, *amazon*, consists of images downloaded from online merchants (`www.amazon.com`). These images are of products shot at medium resolution typically taken in an environment with studio lighting conditions. The *amazon* domain contains 31 categories with an average of 90 images each. The images capture the large intra-class variation of these categories, but typically show the objects only from a canonical viewpoint.

Images from a digital SLR camera: The second domain, *dslr*, consists of images that are captured with a digital SLR camera in realistic environments with natural lighting conditions. The images have high resolution (4288×2848) and low noise. *dslr* has images of the 31 object categories, with 5 different objects for each, in an office environment. Each object was captured with on average 3 images taken from different viewpoints, for a total of 423 images.

Images from a webcam: The third domain, *webcam*, consists of images of the 31 categories recorded with a simple webcam. The images are of low resolution (640×480) and show significant noise and color as well as white balance artifacts. Many current imagers on robotic platforms share a similarly-sized sensor, and therefore also possess these sensing characteristics. The resulting *webcam* dataset contains the same 5 objects per category as in *dslr*, for a total of 795 images.

The database represents several interesting visual domain shifts. It allows us to investigate the adaptation of category models learned on the web to SLR and webcam images, which can be thought of as in situ observations on a robotic platform in a realistic office or home environment. Furthermore, domain transfer between the high-quality DSLR images to low-resolution webcam images allows for a very controlled investigation of category model adaptation, as the same objects were recorded in both domains.

The *Office* dataset images are available together with SURF BoW features that are vector quantized to 800 dimensions. We use these features in all experiments except where explicitly indicated otherwise.

We also use a version of the *Office* dataset, available from [63], which consists of the 10 categories from the *Office* dataset that also appear in *Caltech256*. The same SURF BoW 800-dimensional features are available for the *Caltech256* images.

Large-scale database

We also demonstrate the efficiency of our domain adaptation methods in a large-scale setting (Section 5.7). For this purpose, we consider two domains. The source domain, the *Bing* dataset [16], consists of images obtained using the Bing search engine. In our experiments, we train on 50 source domain examples per category. The target domain is a subset of

the images in the *Caltech-256* benchmark dataset. We vary the number of target domain examples from 5 to 20.

Note that we use the original features (Classeme 2625 dimensional) and train/test splits introduced by [16].

5.7 Evaluation

In the following, we evaluate our methods on the datasets described in the previous section and compare the results to state-of-the-art supervised domain adaptation methods in different domain adaptation scenarios. In particular, we compare against the following methods in the experiments where applicable:

svm$_s$ A support vector machine using source training data.

svm$_t$ A support vector machine using target training data.

hfa A max-margin transform approach that learns a latent common space between source and target as well as a classifier that can be applied to points in that common space [44].

gfk The geodesic flow kernel proposed by [63] applied to all source and target data (including test data). Following [63], we use a 1-nearest neighbor classifier with the geodesic flow kernel.

Standard supervised domain adaptation

In our first set of experiments, we use the 10 category subset of the *Office* database, together with the same 10 categories available from the *Caltech* dataset, to evaluate multi-class accuracy in the standard domain adaptation setting where a few labeled examples are available for all categories in the target domain. We follow the setup of [151] and [63]: 20 training examples for *amazon* source (8 for other source domains) and 3 labeled examples per category for the target domain. We created 20 random train/test splits and averaged the results across them.

The multi-class accuracy for each domain pair is shown in Table 5.2. Our **mmdt** method is the top performing overall, achieving 52.5% accuracy averaged over the 12 domain shifts we explored. This result may be somewhat surprising, because **mmdt** encodes no knowledge of the feature representation, but on shifts where features are homogeneous, still outperforms methods like **gfk** and **symm** which *assume* feature homogeneity. This demonstrates the strength of **mmdt** as a generic domain adaptation approach.

Looking at individual domain shifts, we see that **mmdt** outperforms all other methods in 6 out of the 12 domain shifts. Of the results on the *Office* dataset only (the first 6 rows of Table 5.2), **mmdt** performs the best when either the source or target domain is *amazon*. Because the shift between *amazon* and either of the other two *Office* domains (*dslr* and *webcam*) is much more significant than the shift between *dslr* and *webcam*, as indicated by

	Baselines				Our Methods		
	svm$_s$	**s**vm$_t$	**h**fa	**g**fk	**s**ymm	**a**rc-t	**m**mdt
a → w	33.9 ± 0.7	62.4 ± 0.9	61.8 ± 1.1	58.6 ± 1.0	51.0 ± 1.4	55.7 ± 0.9	64.6 ± 1.2
a → d	35.0 ± 0.8	55.9 ± 0.8	52.7 ± 0.9	50.7 ± 0.8	47.9 ± 1.4	50.2 ± 0.7	56.7 ± 1.3
w → a	35.7 ± 0.4	45.6 ± 0.7	45.9 ± 0.7	44.1 ± 0.4	43.7 ± 0.7	43.4 ± 0.5	47.7 ± 0.9
w → d	66.6 ± 0.7	55.1 ± 0.8	51.7 ± 1.0	70.5 ± 0.7	69.8 ± 1.0	71.3 ± 0.8	67.0 ± 1.1
d → a	34.0 ± 0.3	45.7 ± 0.9	45.8 ± 0.9	45.7 ± 0.6	42.7 ± 0.5	42.5 ± 0.5	46.9 ± 1.0
d → w	74.3 ± 0.5	62.1 ± 0.8	62.1 ± 0.7	76.5 ± 0.5	78.4 ± 0.9	78.3 ± 0.5	74.1 ± 0.8
a → c	35.1 ± 0.3	32.0 ± 0.8	31.1 ± 0.6	36.0 ± 0.5	39.1 ± 0.5	37.0 ± 0.4	36.4 ± 0.8
w → c	31.3 ± 0.4	30.4 ± 0.7	29.4 ± 0.6	31.1 ± 0.6	34.0 ± 0.5	31.9 ± 0.5	32.2 ± 0.8
d → c	31.4 ± 0.3	31.7 ± 0.6	31.0 ± 0.5	32.9 ± 0.5	34.9 ± 0.4	33.5 ± 0.4	34.1 ± 0.8
c → a	35.9 ± 0.4	45.3 ± 0.9	45.5 ± 0.9	44.7 ± 0.8	43.8 ± 0.6	44.1 ± 0.6	49.4 ± 0.8
c → w	30.8 ± 1.1	60.3 ± 1.0	60.5 ± 0.9	63.7 ± 0.8	50.5 ± 1.6	55.9 ± 1.0	63.8 ± 1.1
c → d	35.6 ± 0.7	55.8 ± 0.9	51.9 ± 1.1	57.7 ± 1.1	48.6 ± 1.1	50.6 ± 0.8	56.5 ± 0.9
mean	40.0 ± 0.6	48.5 ± 0.8	47.4 ± 0.8	51.0 ± 0.7	48.7 ± 0.9	49.5 ± 0.6	52.5 ± 1.0

Table 5.2: Multi-class accuracy for the standard supervised domain adaptation setting. All results are from our implementation. When averaged across all domain shifts the reported average value for **gfk** was 51.65 while our implementation had an average of 51.0 ± 0.7. Therefore, the result difference is well within the standard deviation over data splits. Red indicates the best result for each domain split. Blue indicates the group of results that are close to the best-performing result. The domain names are shortened for space: a: *amazon*, w: *webcam*, d: *dslr*, c: *caltech*.

the large performance discrepancy between *amazon* and non-*amazon* shifts with the **svm**$_S$ method, this result indicates that **mmdt** is particularly well-suited to handling larger domain shifts.

Our other methods, **symm** and **arc-t**, have better performance than **mmdt** (and all baselines) on the *webcam* and *dslr* shifts. This demonstrates the utility of these methods in learning smaller domain shifts. Their higher relative performance on such tasks might be due to their cross-domain pairwise constraints on individual examples, which may be less meaningful in cases when the domain shift is larger and individual pairs of examples from a particular category are unlikely to correspond. The **gfk** baseline also performs well on the *webcam* and *dslr* shifts. This fits with our intuition since **gfk** is a 1-nearest neighbor approach and, as such, is more suitable when the domains are initially similar.

In the *caltech* results (the last 6 rows of Table 5.2), we see that the task overall is much easier when *caltech* is the source domain than when it is the target domain, indicating that the *caltech* data is more valuable for recognition in the *Office* domains than the *Office* data is for recognition of the *caltech* categories. When *caltech* is the target domain, the more difficult of the two situations, our **symm** method outperforms all others. On the other hand, when *caltech* is the source domain, we see the best performance from our **mmdt** method and the

source	target	**s**vm$_t$	**hfa**	**arc-t**	**mmdt**
amazon	dslr-600	52.9 ± 0.7	57.8 ± 0.6	58.2 ± 0.6	62.3 ± 0.8
webcam	dslr-600	51.8 ± 0.6	60.0 ± 0.6	58.2 ± 0.7	63.3 ± 0.5

Table 5.3: Multi-class accuracy results on the standard supervised domain adaptation task with different feature dimensions in the source and target. The target domain is *dslr* for both cases.

source	**s**vm$_s$	**g**fk	**a**rc-t	**m**mdt
amazon	10.3 ± 0.6	38.9 ± 0.4	41.4 ± 0.3	44.6 ± 0.3
webcam	51.6 ± 0.5	62.9 ± 0.5	59.4 ± 0.4	58.3 ± 0.5

Table 5.4: Multi-class accuracy results on the Office dataset for the domain shift of *webcam* $\rightarrow$ *dslr* for target test categories not seen at training time.

gfk baseline, with our **arc-t** method performing somewhere in between in most cases. This seems to indicate that **mmdt** is the best of the methods explored when working with a very rich source domain (at least relative to the target domains) like *caltech*, whereas **symm** is superior when the source domain is more homogeneous like the *Office* domains.

Asymmetric features

Next, we analyze the effectiveness of our asymmetric transform learning methods by experimenting with the setting when source and target have different feature dimensions. We use the same experimental setup as previously, but use the full 31 category *Office* dataset and an alternate representation for the *dslr* domain, which is SURF BoW quantized to 600 dimensions (denoted as *dslr-600*). We compare our **mmdt** and **arc-t** methods against **svm**$_t$ and **hfa**. Note that our **symm** method and some baseline methods (**svm**$_s$, **gfk**) are not suited for the asymmetric feature case, as they assume a consistent feature representation across domains. The results are shown in Table 5.3. Again, we find that our **mmdt** method can effectively learn a feature representation for the target domain that optimizes a classification objective. Our **arc-t** method has lower accuracy on this task than **mmdt**, but these results show that it still effectively leverages the source domain data by achieving much higher accuracy than the **svm**$_t$ baseline which ignores the source domain.

Novel categories

We next consider the setting of practical importance where labeled target examples are not available for all objects. Recall that this is a setting that many category specific adaptation methods cannot generalize to, including **hfa** [44] and our **symm** method. Therefore, we

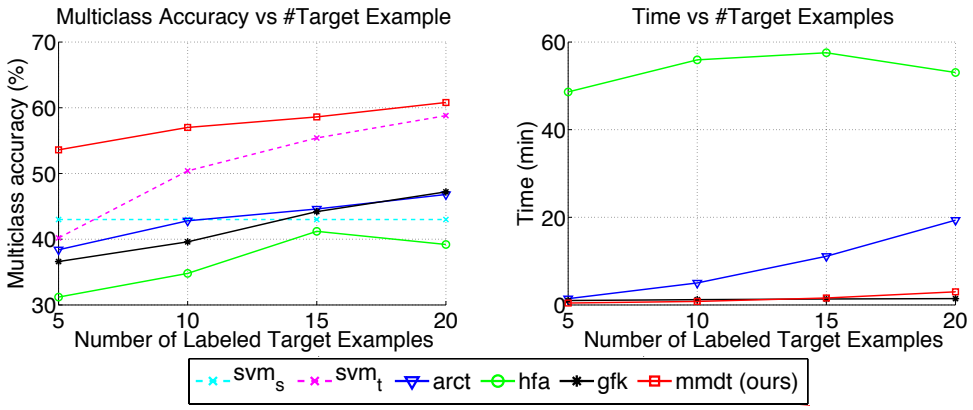


Figure 5.3: Left: multi-class accuracy on the *Bing* dataset using 50 training examples in the source and varying the number of available labeled examples in the target. Right: training time comparison.

compare results from our **mmdt** and **arc-t** methods, which learn category independent feature transforms, to the **gfk** method of [63], which learns a category independent kernel to compare the domains. We use the full *Office* dataset and allow 20 labeled examples per category in the source for *amazon* and 10 labeled examples for the first 15 object categories in the target (*dslr*). For the *webcam* $\rightarrow$ *dslr* shift, we use 8 labeled examples per category in the source for *webcam* and 4 labeled examples for the first 15 object categories in the target *dslr*.

The experimental results for the domain shift of *webcam* $\rightarrow$ *dslr* are evaluated and shown in Table 5.4. **mmdt** outperforms the baselines for the *amazon* $\rightarrow$ *dslr* shift and offers adaptive benefit over $\mathbf{svm}_s$ for the shift from *webcam* $\rightarrow$ *dslr*. As in the first set of experiments, both **arc-t** and **gfk** use nearest neighbor classifiers on a learned kernel which are more suitable to the *webcam* $\rightarrow$ *dslr* shift, as these two domains are initially very similar.

Large-scale data

With our last experiment, we show that our method not only offers high accuracy performance; it also scales well with an increasing dataset size. Specifically, the number of constraints our algorithm optimizes scales linearly with the number of training points. Conversely, the number of constraints that need to be optimized for the **a**rc-t baseline is quadratic in the number of training points.

To demonstrate the effect that constraint set size has on run-time performance, we perform experiments on the *Bing* (source) and *Caltech256* domains described in Section 5.6. The left-hand plot in Figure 11.6 presents multi-class accuracy for this setup. Additionally, the training time of our method and that of the baselines is shown on the right-hand plot.

Our **m**mdt method provides a considerable improvement over **a**rc-t and all the baselines in terms of multi-class accuracy. It is also considerably faster than all but the **g**fk method. Note that **h**fa and **g**fk do not vary significantly as the number of target training points increases. However, for **h**fa the main bottleneck time is consumed by a distance computation between each pair of training points. Therefore, since there are many more source training points than target, adding a few more target points does not significantly increase the overall time spent for this experiment, but would present a problem as the size of the dataset grew in general.

Chapter 6

Extending to Deep Features

6.1 Deep Domain Adaptation

Supervised deep convolutional neural networks (CNNs) trained on large-scale classification tasks have been shown to learn impressive mid-level structures and obtain high levels of performance on contemporary classification challenges [15, 194]. These models generally assume extensive training using labeled data, and testing is limited to data from the same domain. In practice, however, the images we would like to classify are often produced under different imaging conditions or drawn from a different distribution, leading to a domain shift. Scaling such models to new domains remains an open challenge.

Deep CNNs require large amounts of training data to learn good mid-level convolutional models and final fully-connected classifier stages. While the continuing expansion of web-based datasets like ImageNet [15] promises to produce labeled data for almost any desired category, such large-scale supervised datasets may not include images of the category across all domains of practical interest. Earlier deep learning efforts addressed this challenge by learning layers in an unsupervised fashion using unlabeled data to discover salient mid-level structures [29, 34]. While such approaches are appealing, they have heretofore been unable to match the level of performance of supervised models, and unsupervised training of networks with the same level of depth as [103] remains a challenge.

Unfortunately, image datasets are inherently biased [175]. Theoretical [11, 19] and practical results from [151, 175] have shown that supervised methods' test error increases in proportion to the difference between the test and training input distribution. Many visual domain adaptation methods have been put forth to compensate for dataset bias [32, 188, 7, 151, 104, 99, 64, 63, 82, 83], but are limited to shallow models. Evaluation for image category classification across visually distinct domains has focused on the Office dataset, which contains 31 image categories and 3 domains [151]. Recently, [40] showed that using the deep mid-level features learned on ImageNet, instead of the more conventional bag-of-words features, effectively removed the bias in some of the domain adaptation settings in the Office dataset [151]. However, [40] limited their experiments to small-scale source domains

found only in Office, and evaluated on only a subset of relevant layers.

Most prior domain adaptation studies did not use ImageNet as the *source* domain, nor utilize the full set of parameters of a deep CNN trained on source data. Recent work by Rodner et al. [148] attempted to adapt from ImageNet to the SUN dataset, but did not take advantage of deep convolutional features.

We ask the question: will deep models still suffer from dataset bias when trained with all layers of the CNN and a truly large scale source dataset? Here, we provide the first evaluation of domain adaptation with deep learned representations in its most natural setting, in which all of ImageNet is used as source data for a target category. We use the 1.2 million labeled images available in the 2012 ImageNet 1000-way classification dataset [15] to train the model in [103] and evaluate its generalization to the Office dataset. This constitutes a three orders of magnitude increase in source data compared to the several thousand images available for the largest domain in Office.

We find that it is easier to adapt from ImageNet than from previous smaller source domains, but that dataset bias remains a major issue. Fine-tuning the parameters on the small amount of labeled target data (we consider one-shot adaptation) turns out to be unsurprisingly problematic. Instead, we propose a simple yet intuitive adaptation method: train a final domain-adapted classification "layer" using various layers of the pre-trained network as features, without any fine-tuning its parameters. We provide a comprehensive evaluation of existing methods for classifier adaptation as applied to each of the fully connected layers of the network, including the last, task-specific classification layer. When adapting from ImageNet to Office, it turns out to be possible to achieve target domain performance on par with source domain performance using only a single labeled example per target category.

We examine both the setting where there are a few labeled examples from the target domain (*supervised adaptation*) and the setting where there are no labeled target examples (*unsupervised adaptation*). We also describe practical solutions for choosing between the various adaptation methods based on experimental constraints such as limited computation time.

6.2 Background: Deep Domain Adaptation Approaches

For our task we consider adapting between a large source domain and a target domain with few or or no labeled examples. A typical approach to domain adaptation or transfer learning with deep architectures is to take the representation learned via back-propagation on a large dataset, and then transfer the representation to a smaller dataset by fine-tuning, i.e. backpropagation at a lower learning rate [59, 194]. However, fine-tuning requires an ample amount of labeled target data and so should not be expected to work well when we consider the very sparse label condition, such as the *one-shot learning* scenario we evaluate below, where we have just one labeled example per category in the target domain.

In fact, in our experiments under this setting, fine-tuning actually reduces performance. Specifically, on the ImageNet→Webcam task reported in Section 9.4, using the final output layer as a predictor in the target domain received 66% accuracy, while using the final output layer after fine tuning produced a degraded accuracy of 61%.

A separate method that was recently proposed for deep adaptation is called Deep Learning for domain adaptation by Interpolating between Domains (DLID) [25]. This method learns multiple unsupervised deep models directly on the source, target, and combined datasets and uses a representation which is the concatenation of the outputs of each model as its adaptation approach. While this was shown to be an interesting approach, it is limited by its use of unsupervised deep structures.

In general, unsupervised deep convolutional models have been unable to achieve the performance of supervised deep CNNs. However, training a supervised deep model requires sufficient labeled data. Our insight is that the extensive labeled data available in the source domain can be exploited using a supervised model without requiring a significant amount of labeled target data.

Therefore, we propose using a supervised deep source model with supervised or unsupervised adaptation algorithms that are applied to models learned on the target data directly. This hybrid approach will utilize the strong representation available from the supervised deep model trained on a large source dataset while requiring only enough target labeled data to train a shallow model with far fewer parameters. Specifically, we consider training a convolutional neural network (CNN) on the source domain and using that network to extract features on the target data that can then be used to train an auxiliary shallow learner. For extracting features from the deep source model, we follow the setup of Donahue et al. [40], which extracts a visual feature *DeCAF* from the ImageNet-trained architecture of [103].

6.3 Adapting Deep CNNs with Few Labeled Target Examples

We propose a general framework for selectively adapting the parameters of a convolutional neural network (CNN) whose representation and classifier weights are trained on a large-scale source domain, such as ImageNet. Our framework adds a final domain-adaptive classification "layer" that takes the activations of one of the existing network's layers as input features. Note that the network cannot be effectively fine-tuned without access to more labeled target data. This adapted layer is a linear classifier that combines source and target training data using an adaptation method. To demonstrate the generality of our framework, we select a representative set of popular linear classifier adaptation approaches that we empirically evaluate in Section 9.4. We separate our discussion into the set of supervised and unsupervised adaptation settings.

Below we denote the features extracted over the source domain as $\boldsymbol{X}$ and the features extracted over the target domain as $\tilde{\boldsymbol{X}}$. Similarly, we denote the source domain image

classifier as $\boldsymbol{\theta}$ and the target domain image classifier as $\tilde{\boldsymbol{\theta}}$.

Unsupervised Adaptation

Many unsupervised adaptation techniques seek to minimize the distance between subspaces that represent the source and target domains. We denote these subspaces as U and $\tilde{U}$, respectively.

GFK [63] The Geodesic Flow Kernel (GFK) method [63] is an unsupervised domain adaptation approach which seeks embeddings for the source and target points that minimize domain shift. Inputs to the method are U and $\tilde{U}$, lower-dimensional embeddings of the source and target domains (e.g. from principal component analysis). The method constructs the geodesic flow $\phi(t)$ along the manifold of subspaces such that $U = \phi(0)$ and $\tilde{U} = \phi(1)$. Finally, a transformation G is constructed by computing $G = \int_0^1 \phi(t)\phi(t)^\intercal dt$ using a closed-form solution, and classification is performed by training an SVM on the source data $\boldsymbol{X}$ and transformed target data $G\tilde{\boldsymbol{X}}$.

SA [53] The Subspace Alignment (SA) method [53] also begins with low-dimensional embeddings of the source and target domains U and $\tilde{U}$, respectively. It seeks to minimize in M, a transformation matrix, the objective $\|UM - \tilde{U}\|_F^2$. The analytical solution to this objective is $M^* = U^\intercal\tilde{U}$. Given M^*, an SVM is trained on source data $\mathbf{X}$ and transformed target data $UM^*\tilde{U}^\intercal\tilde{\mathbf{X}}$.

Supervised Adaptation

Late Fusion Perhaps the simplest supervised adaptation method is to independently train a source and target classifier and combine the scores of the two to create a final scoring function. We call this approach Late Fusion. It has been explored by many for a simple adaptation approach. Let us denote the score from the source classifier as $\boldsymbol{\theta}_s$ and the score from the target classifier as $\boldsymbol{\theta}_t$. For our experiments we explore two methods of combining these scores, which are described below:

- **M**ax: Produce the scores of both the source and target classifier and simply choose the max of the two as the final score for each example. Therefore, $\boldsymbol{\theta}_{\text{adapt}} = \max(\boldsymbol{\theta}_s, \boldsymbol{\theta}_t)$.
- **L**inear Interpolation: Set the score for a particular example to equal the convex combination of the source and target classifier scores, $\boldsymbol{\theta}_{\text{adapt}} = (1-\alpha)\boldsymbol{\theta}_s + \alpha\boldsymbol{\theta}_t$. This method requires setting a hyperparameter, α, which determines the weights of the source and target classifiers.

Late Fusion has two major advantages: it is easy to implement, and the source classifier it uses may be precomputed to make adaptation very fast. In the case of the linear interpolation combination rule, however, this method can potentially suffer from having a sensitive hyperparameter. We show a hyperparameter analysis in Section 9.4.

Daumé III [32] This simple feature replication method was proposed for domain adaptation by [32]. The method augments feature vectors with a source component, a target component, and a shared component. Each source data point $\mathbf{x}$ is augmented to $\mathbf{x}' = (\mathbf{x}; \mathbf{x}; \mathbf{0})$, and each target data point $\mathbf{v}$ is augmented to $\mathbf{v}' = (\mathbf{v}; \mathbf{0}; \mathbf{v})$. Finally, an SVM is trained on the augmented source and target data—a relatively expensive procedure given the potentially large size of the source domain and the tripled augmented feature dimensionality.

PMT [7] This classifier adaptation method, Projective Model Transfer (PMT), proposed by [7], is a variant of adaptive SVM. It takes as input a classifier $\boldsymbol{\theta}$ pre-trained on the source domain. PMT-SVM learns a target domain classifier $\tilde{\boldsymbol{\theta}}$ by adding an extra term to the usual SVM objective which regularizes the angle $\alpha(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \cos^{-1}\left(\frac{\boldsymbol{\theta}^\intercal \tilde{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}\|\|\tilde{\boldsymbol{\theta}}\|}\right)$ between the target and source hyperplanes. This results in the following loss function:

$$\mathcal{L}_{PMT}(\tilde{\boldsymbol{\theta}}) = \frac{1}{2}\|\tilde{\boldsymbol{\theta}}\|_2^2 + \frac{\Gamma}{2}\|\tilde{\boldsymbol{\theta}}\|_2^2 \sin^2 \alpha(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) + \ell_{hinge}(\tilde{\mathbf{X}}, \tilde{\boldsymbol{y}}; \tilde{\boldsymbol{\theta}}) \ , \tag{6.1}$$

where $\ell_{hinge}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta})$ denotes the SVM hinge loss of a data matrix $\boldsymbol{X}$, label vector $\boldsymbol{Y}$, and classifier hyperplane $\boldsymbol{\theta}$, and Γ is a hyperparameter which, as it increases, enforces more transfer from the source classifier.

MMDT [83] The Max-margin Domain Transforms (MMDT) method from [83] jointly optimizes an SVM-like objective over a feature transformation matrix $\left[A\right]$ mapping target points to the source feature space and classifier parameters $\boldsymbol{\theta}$ in the source feature space. In particular, MMDT minimizes the following loss function (assuming a binary classification task to simplify notation, and with ℓ_{hinge} defined as in PMT):

$$\mathcal{L}_{MMDT}(\boldsymbol{\theta}, \left[A\right]) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\| \left[A\right] - \left[I\right] \|_F^2 + C_s \ell_{hinge}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta}) + C_t \ell_{hinge}(\left[A\right] \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}; \boldsymbol{\theta}) \ , \tag{6.2}$$

where C_s and C_t are hyperparameters controlling the importance of correctly classifying the source and target points (respectively).

6.4 Evaluation with Deep Features

Datasets

The Office [151] dataset is a collection of images from three distinct domains: Amazon, DSLR, and Webcam. The 31 categories in the dataset consist of objects commonly encountered in office settings, such as keyboards, file cabinets, and laptops. Of these 31 categories, 16 overlap with the categories present in the 1000-category ImageNet classification task[1]. Thus, for our experiments, we limit ourselves to these 16 classes. In our experiments using

[1] The 16 overlapping categories are *backpack*, *bike helmet*, *bottle*, *desk lamp*, *desktop computer*, *file cabinet*, *keyboard*, *laptop computer*, *mobile phone*, *mouse*, *printer*, *projector*, *ring binder*, *ruler*, *speaker*, and *trash can*.

Amazon as a source domain, we follow the standard training protocol for this dataset of using 20 source examples per category [151, 63], for a total of 320 images.

ImageNet [15] is the largest available dataset of image category labels. We use 1000 categories' worth of data (1.2M images) to train the network, and use the 16 categories that overlap with Office (approximately 1200 examples per category or ≈20K images total) as labeled source classifier data.

Experimental Setup & Baselines

For our experiments, we use the fully trained deep CNN model described in Section 6.2, extracting feature representations from three different layers of the CNN. We then train a source classifier using these features on one of two source domains, and adapt to the target domain.

The source domains we consider are either the Amazon domain, or the corresponding 16-category ImageNet subset where each category has many more examples. We focus on the Webcam domain as our target (test) domain, as Amazon-to-Webcam was shown to be the only challenging shift in [40] (the DSLR domain is much more similar to Webcam and did not require adaptation when using deep mid-level features). This combination exemplifies the shift from online web images to real-world images taken in typical office/home environments. Note that, regardless of the source domain chosen to learn the classifier, ImageNet data from all 1000 categories was used to train the network.

In addition, for the supervised adaptation setting we assume access to only a single example per category from the target domain (Webcam).

Each method is then evaluated across 20 random train/test splits, and we report averages and standard errors for each setting. For each random train/test split we choose one example for training and 10 other examples for testing (so there is a balanced test set across categories). Therefore, each test split has 160 examples. The unsupervised adaptation methods operate in a transductive setting, so the target subspaces are learned from the unlabeled test data.

Non-adaptive Baselines In addition to the adaptation methods outlined in Section 6.3, we also evaluate using the following non-adaptive baselines.

- **SVM (source only)**: A support vector machine trained only on source data.
- **SVM (target only)**: A support vector machine trained only on target data.
- **SVM (source and target)**: A support vector machine trained on both source and target data. To account for the large discrepancy between the number of training data points in the source and target domains, we weighted the data points such that the constraints from the source and target domains effectively contribute equally to the optimization problem. Specifically, each source data point receives a weight of $\frac{n_t}{n_s+n_t}$,

and each target data point receives a weight of $\frac{n_s}{n_s+n_t}$, where n_s, n_t denote the number of data points in the source and target, respectively.

Many of the adaptation methods we evaluate have hyperparameters that must be cross-validated for use in practice, so we set the parameters of the adaptation techniques as follows.

First, the C value used for C-SVM in the classifier for all methods is set to $C = 1$. Without any validation data we are not able to tune this parameter properly, so we choose to leave it as the default value. Since all methods we report require setting of this parameter, we feel that the relative comparisons between methods is sound even if the absolute numbers could be improved with a new setting for C. For Daumé III and MMDT, which look at the source and target data simultaneously, we use the same weighting scheme as we did for the source and target SVM. Late Fusion with the linear interpolation combination rule is reported across hyperparameter settings in Figure 6.1a to help understand how performance varies as we trade off emphasis between the learned classifiers from the source and target domains. Again, we do not have the validation data to tune this parameter so we report in the tables the performance averaged across parameter settings. The plot vs α indicates that there is usually a best parameter setting that could be learned with more available data. For PMT, we choose $\Gamma = 1000$, which corresponds to allowing a large amount of transfer from the source classifier to the target classifier. We do this because the source-only classifier is stronger than the target-only classifier (with ImageNet source). For the unsupervised methods GFK and SA, again we evaluated a variety of subspace dimensionalities and Figure 6.1b shows that the overall method performance does not vary significantly with the dimensionality choice.

Effect of Source Domain Size

Previous studies considered source domains from the Office dataset. In this section, we ask what happens when an orders-of-magnitude larger source dataset is used.

For completeness we begin by evaluating Amazon as a source domain. Preliminary results on this setting are reported in [40], here we extend the comparison here by presenting the results with more adaptation algorithms and more complete evaluation of hyperparameter settings. Table 6.1 presents multiclass accuracies for each algorithm using either layer 6 or 7 from the deep network, which corresponds to the output from each of the fully connected layers.

An SVM trained using only Amazon data achieves 78.6% in-domain accuracy (tested on the same domain) when using the DeCAF_6 feature and 80.2% in-domain accuracy when using the DeCAF_7 feature. These numbers are significantly higher than the performance of the same classifier on Webcam test data, indicating that even with the DeCAF features, there is a still a domain shift between the Amazon and Webcam datasets.

Next, we consider an unsupervised adaptation setting where no labeled examples are available from the target dataset. In this scenario, we apply two state-of-the-art unsupervised adaptation methods, GFK [63] and SA [53]. Both of these methods make use of a subspace dimensionality hyperparameter. We show the results using a 100-dimensional subspace and

Adaptation Method	Training Data	DeCAF_6	DeCAF_7
SVM (source only)	Amazon	50.28 ± 1.8	**54.08 ± 1.7**
SVM (target only)	Webcam	62.28 ± 1.8	64.97 ± 1.8
GFK [63]	Amazon	**53.13 ± 1.1**	**53.39 ± 1.1**
SA [53]	Amazon	51.74 ± 1.2	**53.86 ± 1.0**
SVM (source and target)	Amazon+Webcam	62.91 ± 1.8	65.82 ± 1.4
Late Fusion (Max)	Amazon+Webcam	65.35 ± 1.7	58.42 ± 1.1
Late Fusion (Lin. Int. Avg)	Amazon+Webcam	63.23 ± 1.4	64.29 ± 1.3
Daumé III [32]	Amazon+Webcam	68.89 ± 1.9	**72.09 ± 1.4**
PMT [7]	Amazon+Webcam	64.84 ± 1.5	65.63 ± 1.8
MMDT [83]	Amazon+Webcam	65.47 ± 1.8	68.10 ± 1.5
Late Fusion (Lin. Int. Oracle)	Amazon+Webcam	71.1 ± 1.7	**72.82 ± 1.4**

Table 6.1: Amazon→Webcam adaptation experiment. We show here multiclass accuracy on the target domain test set for both supervised and unsupervised adaptation experiments across the two fully connected layer features (similar to [40], but with one labeled target example). The best performing unsupervised adaptation algorithms are shown in blue and the best performing supervised adaptation algorithms are shown in red.

leave the discussion of setting this parameter until Section 6.4. For this shift the adaptation algorithms increase performance when using the layer 6 feature, but offer no additional improvement when using the layer 7 feature.

We finally assume that a single example per category is available in the target domain. As the bottom rows of Table 6.1 show, supervised adaptation algorithms are able to provide significant improvement regardless of the feature space chosen, even in the one-shot scenario. For this experiment we noticed that using the second fully connected layer (DeCAF_7) was a stronger overall feature in general.

Adapting with a Large Scale Source Domain

We next address one of the main questions of this paper: Is there still a domain shift when using a large source dataset such as ImageNet? To begin to answer this question we follow the same experimental paradigm as the previous experiment, but use ImageNet as our source dataset. The results are shown in Table 6.2.

Again, we first verify that the source only SVM achieves higher performance when tested on in-domain data than on Webcam data. Indeed, for the 16 overlapping labels, the source SVM produces 62.50% accuracy on ImageNet data using DeCAF_6 features and 74.50% accuracy when using DeCAF_7 features. Compare this to the 54% and 59% for Webcam evaluation and a dataset bias is still clearly evident.

Adaptation Method	Training Data	DeCAF_6	DeCAF_7
SVM (source only)	ImageNet	53.51 ± 1.1	59.15 ± 1.1
SVM (target only)	Webcam	62.28 ± 1.8	64.97 ± 1.8
GFK [63]	ImageNet	65.16 ± 1.1	**67.97 ± 1.4**
SA [53]	ImageNet	59.30 ± 1.4	66.08 ± 1.4
SVM (source and target)	ImageNet+Webcam	56.68 ± 1.2	66.93 ± 1.3
Late Fusion (Max)	ImageNet+Webcam	59.59 ± 1.3	68.86 ± 1.2
Late Fusion (Lin. Int. Avg)	ImageNet+Webcam	60.64 ± 1.3	66.45 ± 1.1
Daumé III [32]	ImageNet+Webcam	59.21 ± 1.7	**71.39 ± 1.5**
PMT [7]	ImageNet+Webcam	66.30 ± 2.1	69.81 ± 1.8
MMDT [83]	ImageNet+Webcam	59.21 ± 1.3	67.75 ± 1.4
Late Fusion (Lin. Int. Oracle)	ImageNet+Webcam	71.65 ± 2.0	**76.76 ± 1.3**

Table 6.2: ImageNet→Webcam adaptation experiment. Comparison of unsupervised and supervised adaptation algorithms on the ImageNet to Webcam domain shift. Results are computed using the outputs of each of the fully connected layers as features. The best supervised adaptation performance is indicated in red and the best unsupervised adaptation performance is highlighted in blue.

Note that when using ImageNet as a source domain, overall performance of all algorithms improves. In addition, unsupervised adaptation approaches are more effective than for the smaller source domain experiment.

Adapting a Pre-trained Classifier to a New Label Set

DeCAF_8 differs from the other DeCAF features in that it constitutes the 1000 activations corresponding to the 1000 labels in the ImageNet classification task. In the CNN proposed by [103], these activations are fed into a softmax unit to compute the label probabilities. We instead experiment with using the DeCAF_8 activations directly as a feature representation, which is akin to training another classifier using the output of the 1000-way CNN classifier.

Table 6.3 shows results for various adaptation techniques using both ImageNet and Amazon as source domains. We use the same setup as before, but instead use DeCAF_8 as the feature representation. The ImageNet results are uniformly better with DeCAF_8 than with DeCAF_6 or DeCAF_7, likely due to the fact that DeCAF_8 was explicitly trained on ImageNet data to effectively discriminate between ImageNet categories. Because it can more effectively classify images from the source domain, it is able to better adapt from the source domain to the target domain.

However, we see a negligible difference in performance for Amazon, with performance actually decreasing with respect to DeCAF_7 for certain adaptation methods. We believe

Adaptation Method	Training Data	Source=ImageNet	Source=Amazon
SVM (source only)	Source	66.23 ± 0.8	53.23 ± 1.6
SVM (target only)	Webcam	63.13 ± 1.9	63.13 ± 1.9
GFK [63]	Source	**68.73 ± 1.1**	54.56 ± 1.2
SA [53]	Source	66.08 ± 1.1	55.98 ± 1.0
SVM (source and target)	Source+Webcam	75.13 ± 1.1	63.20 ± 1.7
Late Fusion (Max)	Source+Webcam	71.77 ± 1.4	62.25 ± 0.8
Late Fusion (LinInt Avg)	Source+Webcam	70.56 ± 1.2	64.56 ± 1.3
Daumé III [32]	Source+Webcam	**77.15 ± 1.1**	70.51 ± 1.7
PMT [7]	Source+Webcam	70.28 ± 1.8	66.77 ± 2.1
MMDT [83]	Source+Webcam	73.96 ± 1.2	66.23 ± 1.4
Late Fusion (Lin. Int. Oracle)	Source+Webcam	**76.61 ± 1.5**	71.49 ± 1.3

Table 6.3: ImageNet→Webcam and Amazon→Webcam adaptation experiments using DeCAF_8, the label activations of the CNN trained on the full ImageNet data. Again, we compare multiclass accuracy of various unsupervised and supervised adaptation methods. The best performing unsupervised adaptation algorithm is shown in blue and the best performing supervised adaptation algorithms are shown in red.

this is because the final activation vector is too specific to the 1000-way ImageNet task, and that DeCAF_7 provides a more general representation that is better suited to the Amazon domain. This, in turn, results in improved adaptation. In general, however, the difference between the various DeCAF representations with Amazon as a source are small enough to be insignificant.

Analysis and Practical Considerations

Our adaptation experiments show that, despite its large size, even ImageNet is not large enough to cover all domains, and that traditional domain adaptation methods go a long way in increasing performance and mitigating the effects of this shift. Depending on the characteristics of the problem at hand, our results suggest different methods may be most suitable.

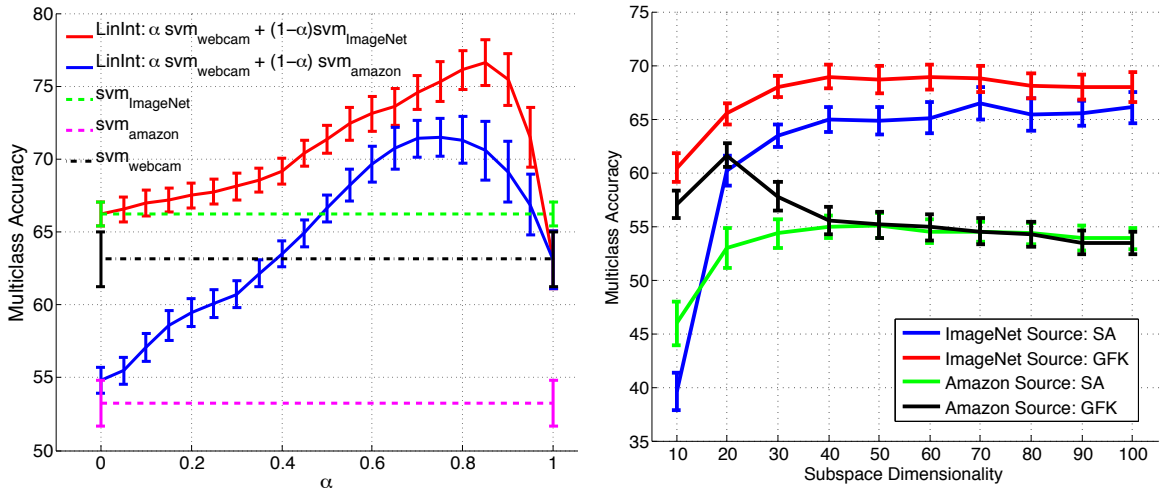
If no labels exist in the target domain, then there are unsupervised adaptation algorithms that are easy to use and fast to compute at adaptation time, yet still achieve increased performance over source-only methods. For this scenario, we experimented with two subspace alignment based methods that both require setting a parameter that indicates the dimensionality of the input subspaces. Figure 6.1b shows the effect that changing the subspace dimensionality has on the overall method performance. In general, we noticed that these methods were not particularly sensitive to this parameter so long as the dimensionality

remains larger than the number of categories in our label set. Below this threshold, the subspace is less likely to capture all important discriminative information needed for classification.

In the case where we have a large source dataset and a limited number of labeled target examples, it may be preferable to compute source classifier parameters in advance, then examine only the source parameters and the target data at adaptation time. Examples of these kinds of methods are Late Fusion and PMT. These methods are unaffected by the number of data points in the source domain at adaptation time, and can thus be applied quickly. In our experiments, we found that a properly tuned Late Fusion classifier with linear interpolation was the fastest and most effective approach. Figure 6.1a shows the performance of linear interpolation Late Fusion as we vary the hyperparameter α. Although the method is sensitive to α, we found that for both source domains, the basic strategy of setting α around 0.8 provides a close approximation to optimal performance. This setting can be interpreted as trusting the target classifier more than the source, but not so much as to completely discount the information available from the source classifier. In each table we report both the performance of linear interpolation both averaged across hyper parameter settings $\alpha \in [0, 1]$ as well as the performance of linear interpolation with the best possible setting of α per experiment – this is denoted as "Oracle" performance.

If there are no computational constraints and there are very few labels in the target domain, the best-performing method seems to be the "frustratingly easy" approach originally proposed by Daumé III [32] and applied again for deep models in [25].

Finally, we found that feature representation can have a significant impact on adaptation performance. Our results show that ImageNet as source performs best with the DeCAF_8 representation, whereas Amazon as source performs best with the DeCAF_7 representation. This, combined with our intuition, seems to indicate that for adaptation from source domains other than ImageNet, an intermediate representation other than DeCAF_8 is more powerful for adaptation, whereas ImageNet classification works best with the full representation that was trained on it.



(a) Late Fusion with Linear Interpolation

(b) Unsupervised Methods

Figure 6.1: Evaluation of hyperparameters for domain adaptation methods. (a) Analysis of the combination hyperparameter α for Late Fusion with linear interpolation. (b) Analysis of the subspace dimensionality for the unsupervised adaptation algorithms

Chapter 7

Summary

We have presented a unified framework for learning a category invariant transformation that has been proven effective for visual domain adaptation. In particular, we derive two specific formulations from the general framework, one which is most useful for learning a similarity function between a source and target domain independent of the classifier, and another which focuses on learning linear classifiers in a max-margin framework.

We demonstrated the importance of using a domain adaptation method to boost overall performance for visual recognition tasks, and analyze the scenarios in which a max-margin objective and a transformation-based approach are most beneficial. In our experiments, we provided an in-depth analysis and comparison of the different algorithms we presented and their connection to other state-of-the-art methods.

In the future, we would like to extend further to a multi-domain scenario, where lots of labeled and heterogenous source data can be exploited to help classification in a target domain.

Further, we presented the first evaluation of domain adaptation from a large-scale source dataset with deep features. We demonstrated that, although using ImageNet as a source domain generalizes better than other smaller source domains, there is still a domain shift when adapting to other visual domains.

Our experimental results show that deep adaptation methods can go a long way in mitigating the effects of this domain shift. Based on our results, we also provided a set of practical recommendations for choosing a feature representation and adaptation method accounting for constraints on runtime and accuracy.

There are a number of interesting directions to take given our results. First we notice that though DeCAF_8 is the strongest feature to use for learning a classifier on ImageNet data, DeCAF_7 is actually a better feature to use with the Amazon source domain and the Webcam target domain. This could lead to a hybrid approach where one uses different feature representations for the various domains and produces a combined adapted model. Another interesting direction that should be explored is to integrate the adaption algorithms into the deep models explicitly and even allow for feedback between the two stages. Current deep models although allow information flow between the final classifier and the representation

learning architecture. We feel that the next step is to have a separate task specific adaptable layer that does not simply learn a new final layer, but instead learns a separate, but equivalent final layer, that is regularized by the final layer learned on the source dataset.

This future work is a natural extension of the result we have shown in this paper: that pre-trained deep representations with large source domains can be effectively adapted to new target domains using only shallow, linear adaptation methods, and that in cases where the target data is limited, this approach is the best way to mitigate dataset bias.

Part III

Domain Invariant Representation Learning with Deep Models

Chapter 8

Deep Models for Visual Recognition

In recent years, convolutional neural networks have emerged as the dominant paradigm for visual recognition. The high capacity models along with large amounts of annotated images for supervised training on modern GPU hardware has facilitated the rapid resurgence in the use of these networks. However, it is important to state that convolutional networks and more generally neural networks are not a newly developed technology.

The earliest approaches using a neural network architecture are often attributed to the infamous neurological experiment studying the visual cortex of cats by Hubel and Wiesel [88] and the simultaneous invention of the perceptron [149, 131]. Following, these works, convolutional neural networks were first introduced in the neocognitron by Fukushima [55] for hand-writing digit recognition and later popularized with the efficient training procedure of back-propagation by Lecun *et al.* [111, 129].

While these approaches provided the basic research behind neural networks and convolutional networks for visual representations, it was the work of Krizhevsky *et al.* [103], which provided the engineered deep architecture along with a training procedure on the large scale ImageNet [35] dataset, which first introduced a competitive model for object classification.

Since that time, a large body of research has been devoted to extending the use of convolutional networks for other recognition tasks such as detection [152, 59] and semantic segmentation [123, 192, 122].

In this part, we consider an adaptation approach for object classification with deep models. In particular, we take advantage of the seamless nature of the image representation and object classification model within a convolutional network and in the next chapter introduce the notion of optimizing for domain invariance (or minimizing the distance between the source and target representations), while simultaneously maintaining a strong object recognition model.

Chapter 9

Optimize for Domain Invariance

9.1 Introduction

Consider a group of robots trained by the manufacturer to recognize thousands of common objects using standard image databases, then shipped to households around the country. As each robot starts to operate in its own unique environment, it is likely to have degraded performance due to the shift in domain. It is clear that, given enough extra supervised data from the new environment, the original performance could be recovered. However, state-of-the-art recognition algorithms rely on high capacity convolutional neural network (CNN) models that require millions of supervised images for initial training. Even the traditional approach for adapting deep models, fine-tuning [59, 152], may require hundreds or thousands of labeled examples for each object category that needs to be adapted.

It is reasonable to assume that the robot's new owner will label a handful of examples for a few types of objects, but completely unrealistic to presume full supervision in the new environment. Therefore, we propose an algorithm that effectively adapts between the training (source) and test (target) environments by utilizing both generic statistics from unlabeled data collected in the new environment as well as a few human labeled examples from a subset of the categories of interest. Our approach performs transfer learning both across domains and across tasks (see Figure 9.1). Intuitively, domain transfer is accomplished by making the marginal feature distributions of source and target as similar to each other as possible. Task transfer is enabled by transferring empirical category correlations learned on the source to the target domain. This helps to preserve relationships between categories, e.g., *bottle* is similar to *mug* but different from *keyboard*. Previous work proposed techniques for domain transfer with CNN models [57, 124] but did not utilize the learned source semantic structure for task transfer.

To enable domain transfer, we use the unlabeled target data to compute an estimated marginal distribution over the new environment and explicitly optimize a feature representation that minimizes the distance between the source and target domain distributions. Dataset bias was classically illustrated in computer vision by the "name the dataset" game

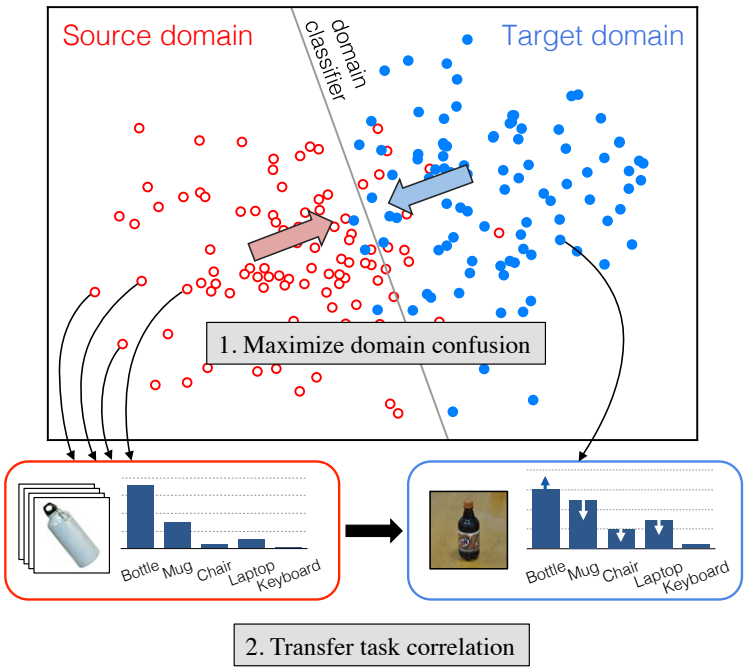


Figure 9.1: We transfer discriminative category information from a source domain to a target domain via two methods. First, we maximize domain confusion by making the marginal distributions of the two domains as similar as possible. Second, we transfer correlations between classes learned on the source examples directly to the target examples, thereby preserving the relationships between classes.

of Torralba and Efros [175], which trained a classifier to predict which dataset an image originates from, thereby showing that visual datasets are biased samples of the visual world. Indeed, this turns out to be formally connected to measures of domain discrepancy [100, 20]. Optimizing for domain invariance, therefore, can be considered equivalent to the task of learning to predict the class labels while simultaneously finding a representation that makes the domains appear as similar as possible. This principle forms the domain transfer component of our proposed approach. We learn deep representations by optimizing over a loss which includes both classification error on the labeled data as well as a *domain confusion* loss which seeks to make the domains indistinguishable.

However, while maximizing domain confusion pulls the marginal distributions of the domains together, it does not necessarily align the classes in the target with those in the source. Thus, we also explicitly transfer the similarity structure amongst categories from the source to the target and further optimize our representation to produce the same structure in the target domain using the few target labeled examples as reference points. We are inspired by prior work on distilling deep models [8, 74] and extend the ideas presented in these works to a domain adaptation setting. We first compute the average output probability

distribution, or "soft label," over the source training examples in each category. Then, for each target labeled example, we directly optimize our model to match the distribution over classes to the soft label. In this way we are able to perform task adaptation by transferring information to categories with no explicit labels in the target domain.

We solve the two problems jointly using a new CNN architecture, outlined in Figure 19.2. We combine a domain confusion and softmax cross-entropy losses to train the network with the target data. Our architecture can be used to solve *supervised adaptation*, when a small amount of target labeled data is available from each category, and *semi-supervised adaptation*, when a small amount of target labeled data is available from a subset of the categories. We provide a comprehensive evaluation on the popular Office benchmark [151] and the recently introduced cross-dataset collection [174] for classification across visually distinct domains. We demonstrate that by jointly optimizing for domain confusion and matching soft labels, we are able to outperform the current state-of-the-art visual domain adaptation results.

9.2 Related work

There have been many approaches proposed in recent years to solve the visual domain adaptation problem, which is also commonly framed as the visual dataset bias problem [175]. All recognize that there is a shift in the distribution of the source and target data representations. In fact, the size of a domain shift is often measured by the distance between the source and target subspace representations [20, 53, 100, 126, 138]. A large number of methods have sought to overcome this difference by learning a feature space transformation to align the source and target representations [151, 104, 53, 63]. For the *supervised* adaptation scenario, when a limited amount of labeled data is available in the target domain, some approaches have been proposed to learn a target classifier regularized against the source classifier [188, 7, 16]. Others have sought to both learn a feature transformation and regularize a target classifier simultaneously [83, 40].

Recently, supervised CNN based feature representations have been shown to be extremely effective for a variety of visual recognition tasks [103, 40, 59, 152]. In particular, using deep representations dramatically reduces the effect of resolution and lighting on domain shifts [40, 76]. Parallel CNN architectures such as Siamese networks have been shown to be effective for learning invariant representations [23, 26]. However, training these networks requires labels for each training instance, so it is unclear how to extend these methods to unsupervised or semi-supervised settings. Multimodal deep learning architectures have also been explored to learn representations that are invariant to different input modalities [134]. However, this method operated primarily in a generative context and therefore did not leverage the full representational power of supervised CNN representations.

Training a joint source and target CNN architecture was proposed by [25], but was limited to two layers and so was significantly outperformed by the methods which used a deeper architecture [103], pre-trained on a large auxiliary data source (ex: ImageNet [15]). [58] proposed pre-training with a denoising auto-encoder, then training a two-layer network

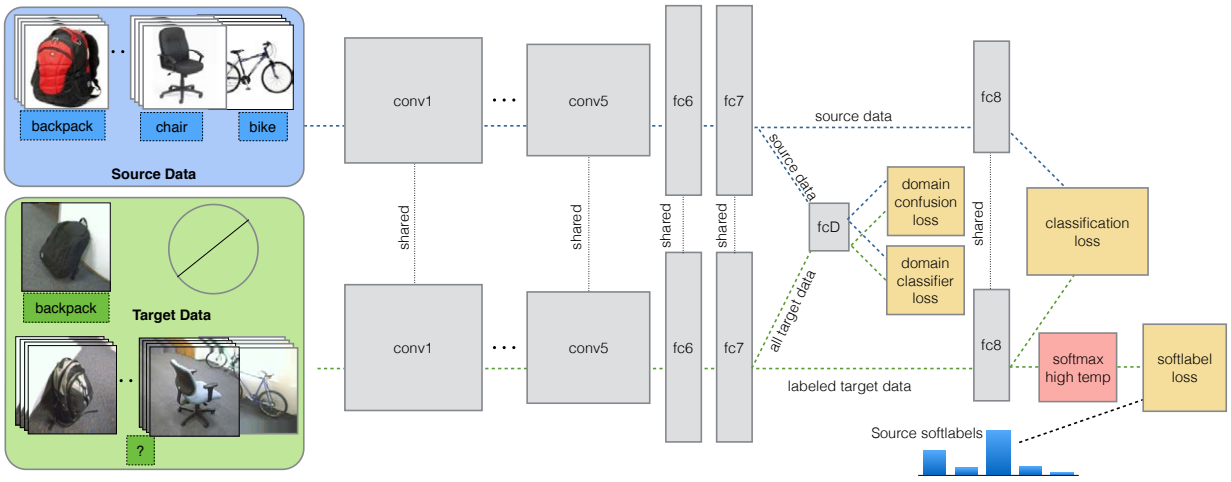


Figure 9.2: Our overall CNN architecture for domain and task transfer. We use a domain confusion loss over all source and target (both labeled and unlabeled) data to learn a domain invariant representation. We simultaneously transfer the learned source semantic structure to the target domain by optimizing the network to produce activation distributions that match those learned for source data in the source only CNN. *Best viewed in color.*

simultaneously with the MMD domain confusion loss. This effectively learns a domain invariant representation, but again, because the learned network is relatively shallow, it lacks the strong semantic representation that is learned by directly optimizing a classification objective with a supervised deep CNN.

Using classifier output distributions instead of category labels during training has been explored in the context of model compression or distillation [8, 74]. However, we are the first to apply this technique in a domain adaptation setting in order to transfer class correlations between domains.

Other works have contemporaneously explored the idea of directly optimizing a representation for domain invariance [57, 124]. However, they either use weaker measures of domain invariance or make use of optimization methods that are less robust than our proposed method, and they do not attempt to solve the task transfer problem in the semi-supervised setting.

9.3 Joint CNN architecture for domain and task transfer

We first give an overview of our convolutional network (CNN) architecture, depicted in Figure 19.2, that learns a representation which both aligns visual domains and transfers the semantic structure from a well labeled source domain to the sparsely labeled target domain.

We assume access to a limited amount of labeled target data, potentially from only a subset of the categories of interest. With limited labels on a subset of the categories, the traditional domain transfer approach of fine-tuning on the available target data [59, 152, 85] is not effective. Instead, since the source labeled data shares the label space of our target domain, we use the source data to guide training of the corresponding classifiers.

Our method takes as input the labeled source data $\{\mathbf{x}, y\}$ (blue box Figure 19.2) and the target data $\{\mathbf{v}, g\}$ (green box Figure 19.2), where the labels g are only provided for a subset of the target examples. Our goal is to produce a category classifier θ_C that operates on an image feature representation $f(x; \theta_{\text{repr}})$ parameterized by representation parameters θ_{repr} and can correctly classify target examples at test time.

For a setting with K categories, let our desired classification objective be defined as the standard softmax loss

$$\mathcal{L}_C(x, y; \theta_{\text{repr}}, \theta_C) = -\sum_k \mathbb{1}[y = k] \log p_k \tag{9.1}$$

where p is the softmax of the classifier activations, $p = \text{softmax}(\theta_C^T f(x; \theta_{\text{repr}}))$.

We could use the available source labeled data to train our representation and classifier parameters according to Equation (9.1), but this often leads to overfitting to the source distribution, causing reduced performance at test time when recognizing in the target domain. However, we note that if the source and target domains are very similar then the classifier trained on the source will perform well on the target. In fact, it is sufficient for the source and target data to be similar under the learned representation, θ_{repr}.

Inspired by the "name the dataset" game of Torralba and Efros [175], we can directly train a domain classifier θ_D to identify whether a training example originates from the source or target domain given its feature representation. Intuitively, if our choice of representation suffers from domain shift, then they will lie in distinct parts of the feature space, and a classifier will be able to easily separate the domains. We use this notion to add a new *domain confusion* loss $\mathcal{L}_{\text{conf}}(\mathbf{x}, \mathbf{v}, \theta_D; \theta_{\text{repr}})$ to our objective and directly optimize our representation so as to minimize the discrepancy between the source and target distributions. This loss is described in more detail in Section 9.3.

Domain confusion can be applied to learn a representation that aligns source and target data without any target labeled data. However, we also presume a handful of sparse labels in the target domain, g. In this setting, a simple approach is to incorporate the target labeled data along with the source labeled data into the classification objective of Equation (9.1)[1]. However, fine-tuning with hard category labels limits the impact of a single training example, making it hard for the network to learn to generalize from the limited labeled data. Additionally, fine-tuning with hard labels is ineffective when labeled data is available for only a subset of the categories.

For our approach, we draw inspiration from recent network distillation works [8, 74], which demonstrate that a large network can be "distilled" into a simpler model by replacing

[1]We present this approach as one of our baselines.

the hard labels with the softmax activations from the original large model. This modification proves to be critical, as the distribution holds key information about the relationships between categories and imposes additional structure during the training process. In essence, because each training example is paired with an output distribution, it provides valuable information about not only the category it belongs to, but also each other category the classifier is trained to recognize.

Thus, we propose using the labeled target data to optimize the network parameters through a *soft label* loss, $\mathcal{L}_{\text{soft}}(\mathbf{v}, g; \theta_{\text{repr}}, \theta_C)$. This loss will train the network parameters to produce a "soft label" activation that matches the average output distribution of source examples on a network trained to classify source data. This loss is described in more detail in Section 9.3. By training the network to match the expected source output distributions on target data, we transfer the learned inter-class correlations from the source domain to examples in the target domain. This directly transfers useful information from source to target, such as the fact that *bookshelves* appear more similar to *filing cabinets* than to *bicycles*.

Our full method then minimizes the joint loss function

$$\begin{aligned}\mathcal{L}(\mathbf{x}, y, \mathbf{v}, g, \theta_D; \theta_{\text{repr}}, \theta_C) = \\ \mathcal{L}_C(\mathbf{x}, y, \mathbf{v}, g; \theta_{\text{repr}}, \theta_C) \\ + \lambda \mathcal{L}_{\text{conf}}(\mathbf{x}, \mathbf{v}, \theta_D; \theta_{\text{repr}}) \\ + \nu \mathcal{L}_{\text{soft}}(\mathbf{v}, g; \theta_{\text{repr}}, \theta_C).\end{aligned} \tag{9.2}$$

where the hyperparameters λ and ν determine how strongly domain confusion and soft labels influence the optimization.

Our ideas of domain confusion and soft label loss for task transfer are generic and can be applied to any CNN classification architecture. For our experiments and for the detailed discussion in this paper we modify the standard Krizhevsky architecture [103], which has five convolutional layers (conv1–conv5) and three fully connected layers (fc6–fc8). The representation parameter θ_{repr} corresponds to layers 1–7 of the network, and the classification parameter θ_C corresponds to layer 8. For the remainder of this section, we provide further details on our novel loss definitions and the implementation of our model.

Aligning domains via domain confusion

In this section we describe in detail our proposed *domain confusion* loss objective. Recall that we introduce the domain confusion loss as a means to learn a representation that is domain invariant, and thus will allow us to better utilize a classifier trained using the labeled source data. We consider a representation to be domain invariant if a classifier trained using that representation can not distinguish examples from the two domains.

To this end, we add an additional domain classification layer, denoted as fcD in Figure 19.2, with parameters θ_D. This layer simply performs binary classification using the domain corresponding to an image as its label. For a particular feature representation, θ_{repr},

we evaluate its domain invariance by learning the best domain classifier on the representation. This can be learned by optimizing the following objective, where y_D denotes the domain that the example is drawn from:

$$\mathcal{L}_D(\mathbf{x}, \mathbf{v}, \theta_{\text{repr}}; \theta_D) = -\sum_d \mathbb{1}[y_D = d] \log q_d \tag{9.3}$$

with q corresponding to the softmax of the domain classifier activation: $q = \text{softmax}(\theta_D^T f(x; \theta_{\text{repr}}))$.

For a particular domain classifier, θ_D, we can now introduce our loss which seeks to "maximally confuse" the two domains by computing the cross entropy between the output predicted domain labels and a uniform distribution over domain labels:

$$\mathcal{L}_{\text{conf}}(\mathbf{x}, \mathbf{v}, \theta_D; \theta_{\text{repr}}) = -\sum_d \frac{1}{D} \log q_d. \tag{9.4}$$

This domain confusion loss seeks to learn domain invariance by finding a representation in which the best domain classifier performs poorly.

Ideally, we want to simultaneously minimize Equations (9.3) and (9.4) for the representation and the domain classifier parameters. However, the two losses stand in direct opposition to one another: learning a fully domain invariant representation means the domain classifier must do poorly, and learning an effective domain classifier means that the representation is not domain invariant. Rather than globally optimizing θ_D and θ_{repr}, we instead perform iterative updates for the following two objectives given the fixed parameters from the previous iteration:

$$\min_{\theta_D} \mathcal{L}_D(\mathbf{x}, \mathbf{v}, \theta_{\text{repr}}; \theta_D) \tag{9.5}$$

$$\min_{\theta_{\text{repr}}} \mathcal{L}_{\text{conf}}(\mathbf{x}, \mathbf{v}, \theta_D; \theta_{\text{repr}}). \tag{9.6}$$

These losses are readily implemented in standard deep learning frameworks, and after setting learning rates properly so that Equation (9.5) only updates θ_D and Equation (9.6) only updates θ_{repr}, the updates can be performed via standard backpropagation. Together, these updates ensure that we learn a representation that is domain invariant.

Aligning source and target classes via soft labels

While training the network to confuse the domains acts to align their marginal distributions, there are no guarantees about the alignment of classes between each domain. To ensure that the relationships between classes are preserved across source and target, we fine-tune the network against "soft labels" rather than the image category hard label.

We define a soft label for category k as the average over the softmax of all activations of source examples in category k, depicted graphically in Figure 9.3, and denote this average as $l^{(k)}$. Note that, since the source network was trained purely to optimize a classification

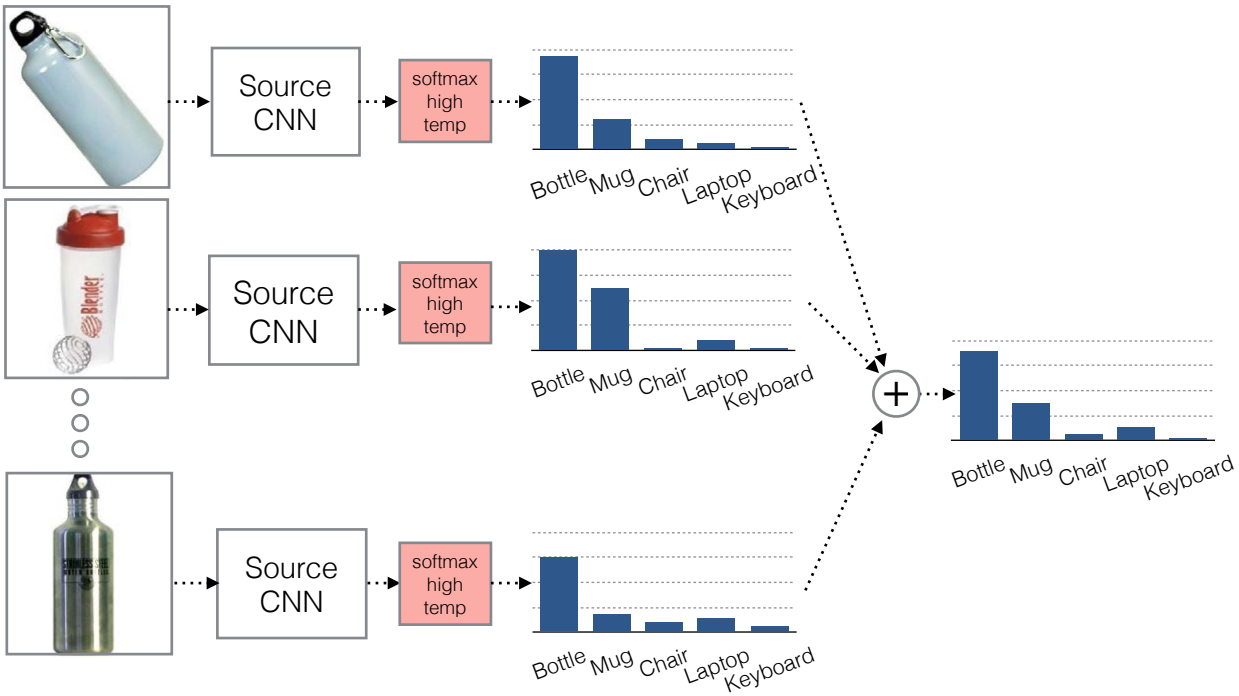


Figure 9.3: Soft label distributions are learned by averaging the per-category activations of source training examples using the source model. An example, with 5 categories, depicted here to demonstrate the final soft activation for the bottle category will be primarily dominated by bottle and mug with very little mass on chair, laptop, and keyboard.

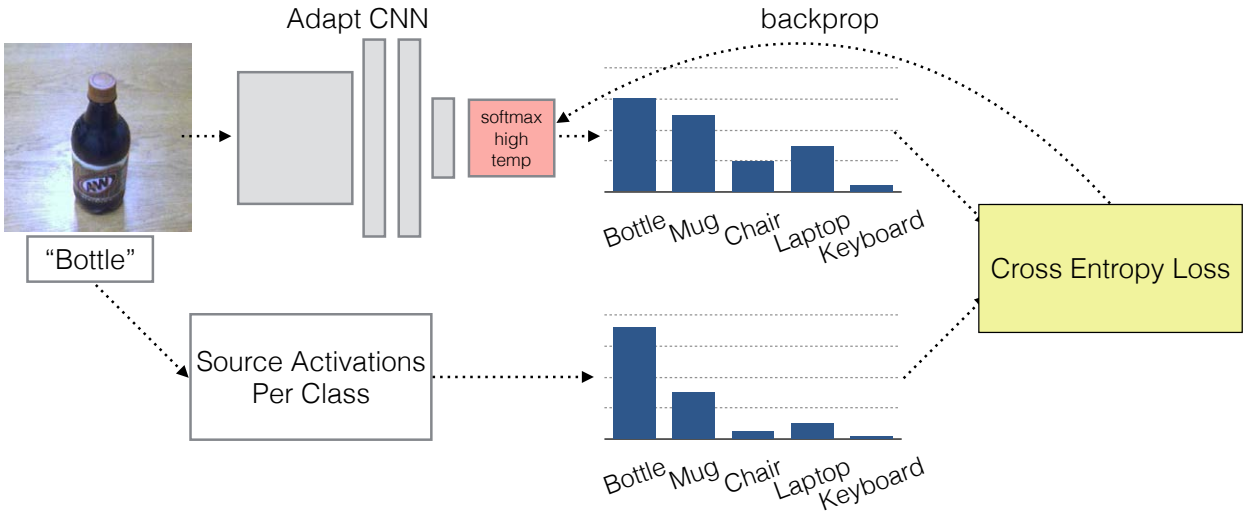


Figure 9.4: Depiction of the use of source per-category soft activations with the cross entropy loss function over the current target activations.

objective, a simple softmax over each z_S^i will hide much of the useful information by producing a very peaked distribution. Instead, we use a softmax with a high temperature τ so that the related classes have enough probability mass to have an effect during fine-tuning. With our computed per-category soft labels we can now define our soft label loss:

$$\mathcal{L}_{\text{soft}}(\mathbf{v}, g; \theta_{\text{repr}}, \theta_C) = -\sum_i l_i^{(g)} \log p_i \tag{9.7}$$

where p denotes the soft activation of the target image, $p = \text{softmax}(\theta_C^T f(\mathbf{v}; \theta_{\text{repr}})/\tau)$. The loss above corresponds to the cross-entropy loss between the soft activation of a particular target image and the soft label corresponding to the category of that image, as shown in Figure 9.4.

To see why this will help, consider the soft label for a particular category, such as *bottle*. The soft label $l^{(\text{bottle})}$ is a K-dimensional vector, where each dimension indicates the similarity of bottles to each of the K categories. In this example, the bottle soft label will have a higher weight on *mug* than on *keyboard*, since bottles and mugs are more visually similar. Thus, soft label training with this particular soft label directly enforces the relationship that bottles and mugs should be closer in feature space than bottles and keyboards.

One important benefit of using this soft label loss is that we ensure that the parameters for categories without any labeled target data are still updated to output non-zero probabilities. We explore this benefit in Section 9.4, where we train a network using labels from a subset of the target categories and find significant performance improvement even when evaluating only on the unlabeled categories.

9.4 Evaluation

To analyze the effectiveness of our method, we evaluate it on the Office dataset, a standard benchmark dataset for visual domain adaptation, and on a new large-scale cross-dataset domain adaptation challenge.

Adaptation on the Office dataset

The Office dataset is a collection of images from three distinct domains, Amazon, DSLR, and Webcam, the largest of which has 2817 labeled images [151]. The 31 categories in the dataset consist of objects commonly encountered in office settings, such as keyboards, file cabinets, and laptops.

We evaluate our method in two different settings:

- **Supervised adaptation** Labeled training data for all categories is available in source and sparsely in target.
- **Semi-supervised adaptation (task adaptation)** Labeled training data is available in source and sparsely for a subset of the target categories.

	$A \to W$	$A \to D$	$W \to A$	$W \to D$	$D \to A$	$D \to W$	Average
DLID [25]	51.9	–	–	89.9	–	78.2	–
$DeCAF_6$ S+T [40]	80.7 ± 2.3	–	–	–	–	94.8 ± 1.2	–
DaNN [58]	53.6 ± 0.2	–	–	83.5 ± 0.0	–	71.2 ± 0.0	–
Source CNN	56.5 ± 0.3	64.6 ± 0.4	42.7 ± 0.1	93.6 ± 0.2	47.6 ± 0.1	92.4 ± 0.3	66.22
Target CNN	80.5 ± 0.5	81.8 ± 1.0	59.9 ± 0.3	81.8 ± 1.0	59.9 ± 0.3	80.5 ± 0.5	74.05
Source+Target CNN	82.5 ± 0.9	85.2 ± 1.1	**65.2 ± 0.7**	96.3 ± 0.5	65.8 ± 0.5	93.9 ± 0.5	81.50
Ours: dom confusion only	**82.8 ± 0.9**	85.9 ± 1.1	64.9 ± 0.5	97.5 ± 0.2	**66.2 ± 0.4**	95.6 ± 0.4	82.13
Ours: soft labels only	82.7 ± 0.7	84.9 ± 1.2	**65.2 ± 0.6**	**98.3 ± 0.3**	66.0 ± 0.5	**95.9 ± 0.6**	82.17
Ours: dom confusion+soft labels	82.7 ± 0.8	**86.1 ± 1.2**	65.0 ± 0.5	97.6 ± 0.2	**66.2 ± 0.3**	95.7 ± 0.5	**82.22**

Table 9.1: Multi-class accuracy evaluation on the standard supervised adaptation setting with the *Office* dataset. We evaluate on all 31 categories using the standard experimental protocol from [151]. Here, we compare against three state-of-the-art domain adaptation methods as well as a CNN trained using only source data, only target data, or both source and target data together.

	$A \to W$	$A \to D$	$W \to A$	$W \to D$	$D \to A$	$D \to W$	Average
MMDT [83]	–	44.6 ± 0.3	–	58.3 ± 0.5	–	–	–
Source CNN	54.2 ± 0.6	63.2 ± 0.4	34.7 ± 0.1	94.5 ± 0.2	36.4 ± 0.1	89.3 ± 0.5	62.0
Ours: dom confusion only	55.2 ± 0.6	63.7 ± 0.9	**41.1 ± 0.0**	96.5 ± 0.1	41.2 ± 0.1	**91.3 ± 0.4**	64.8
Ours: soft labels only	56.8 ± 0.4	65.2 ± 0.9	38.8 ± 0.4	96.5 ± 0.2	41.7 ± 0.3	89.6 ± 0.1	64.8
Ours: dom confusion+soft labels	**59.3 ± 0.6**	**68.0 ± 0.5**	40.5 ± 0.2	**97.5 ± 0.1**	**43.1 ± 0.2**	90.0 ± 0.2	**66.4**

Table 9.2: Multi-class accuracy evaluation on the standard semi-supervised adaptation setting with the *Office* dataset. We evaluate on 16 held-out categories for which we have no access to target labeled data. We show results on these unsupervised categories for the source only model, our model trained using only soft labels for the 15 auxiliary categories, and finally using domain confusion together with soft labels on the 15 auxiliary categories.

For all experiments we initialize the parameters of conv1–fc7 using the released CaffeNet [92] weights. We then further fine-tune the network using the source labeled data in order to produce the soft label distributions and use the learned source CNN weights as the initial parameters for training our method. All implementations are produced using the open source Caffe [92] framework, and the network definition files and cross entropy loss layer needed for training will be released upon acceptance. We optimize the network using a learning rate of 0.001 and set the hyper-parameters to $\lambda = 0.01$ (confusion) and $\nu = 0.1$ (soft).

For each of the six domain shifts, we evaluate across five train/test splits, which are generated by sampling examples from the full set of images per domain. In the source domain, we follow the standard protocol for this dataset and generate splits by sampling 20 examples per category for the Amazon domain, and 8 examples per category for the DSLR and Webcam domains.

We first present results for the supervised setting, where 3 labeled examples are provided

for each category in the target domain. We report accuracies on the remaining unlabeled images, following the standard protocol introduced with the dataset [151]. In addition to a variety of baselines, we report numbers for both soft label fine-tuning alone as well as soft labels with domain confusion in Table 9.1. Because the Office dataset is imbalanced, we report multi-class accuracies, which are obtained by computing per-class accuracies independently, then averaging over all 31 categories.

We see that fine-tuning with soft labels or domain confusion provides a consistent improvement over hard label training in 5 of 6 shifts. Combining soft labels with domain confusion produces marginally higher performance on average. This result follows the intuitive notion that when enough target labeled examples are present, directly optimizing for the joint source and target classification objective (Source+Target CNN) is a strong baseline and so using either of our new losses adds enough regularization to improve performance.

Next, we experiment with the semi-supervised adaptation setting. We consider the case in which training data and labels are available for some, but not all of the categories in the target domain. We are interested in seeing whether we can transfer information learned from the labeled classes to the unlabeled classes.

To do this, we consider having 10 target labeled examples per category from only 15 of the 31 total categories, following the standard protocol introduced with the *Office* dataset [151]. We then evaluate our classification performance on the remaining 16 categories for which no data was available at training time.

In Table 9.2 we present multi-class accuracies over the 16 held-out categories and compare our method to a previous domain adaptation method [83] as well as a source-only trained CNN. Note that, since the performance here is computed over only a subset of the categories in the dataset, the numbers in this table should not be directly compared to the supervised setting in Table 9.1.

We find that all variations of our method (only soft label loss, only domain confusion, and both together) outperform the baselines. Contrary to the fully supervised case, here we note that both domain confusion and soft labels contribute significantly to the overall performance improvement of our method. This stems from the fact that we are now evaluating on categories which lack labeled target data, and thus the network can not implicitly enforce domain invariance through the classification objective alone. Separately, the fact that we get improvement from the soft label training on related tasks indicates that information is being effectively transferred between tasks.

In Figure 9.5, we show examples for the Amazon→Webcam shift where our method correctly classifies images from held out object categories and the baseline does not. We find that our method is able to consistently overcome error cases, such as the notebooks that were previously confused with letter trays, or the black mugs that were confused with black computer mice.

Figure 9.5: Examples from the Amazon→Webcam shift in the semi-supervised adaptation setting, where our method (the bottom turquoise label) correctly classifies images while the baseline (the top purple label) does not.

Adaptation between diverse domains

For an evaluation with larger, more distinct domains, we test on the recent testbed for cross-dataset analysis [174], which collects images from classes shared in common among computer vision datasets. We use the dense version of this testbed, which consists of 40 categories shared between the ImageNet, Caltech-256, SUN, and Bing datasets, and evaluate specifically with ImageNet as source and Caltech-256 as target.

We follow the protocol outlined in [174] and generate 5 splits by selecting 5534 images from ImageNet and 4366 images from Caltech-256 across the 40 shared categories. Each split is then equally divided into a train and test set. However, since we are most interested in evaluating in the setting with limited target data, we further subsample the target training set into smaller sets with only 1, 3, and 5 labeled examples per category.

Results from this evaluation are shown in Figure 9.6. We compare our method to both

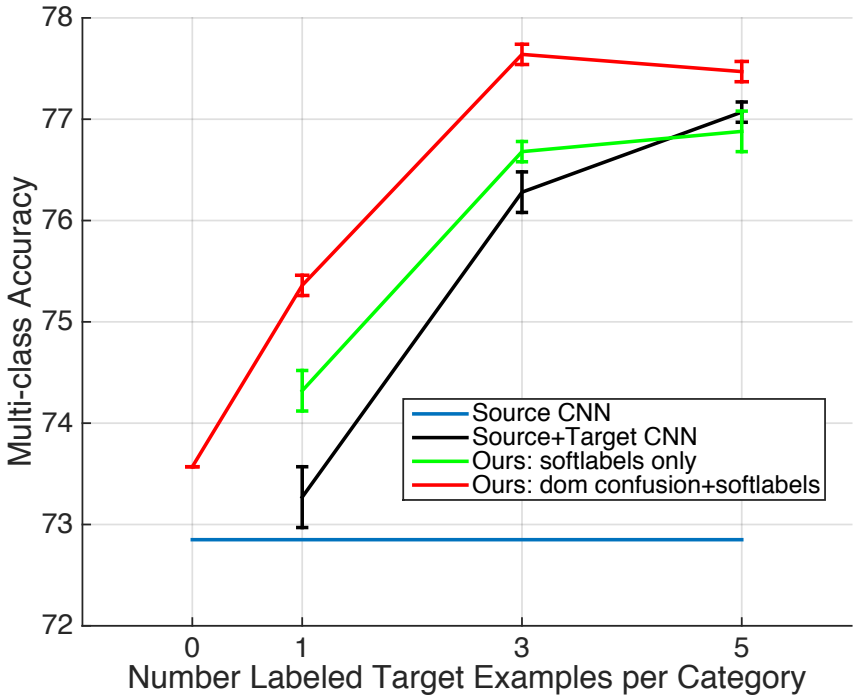


Figure 9.6: ImageNet→Caltech supervised adaptation from the Cross-dataset [174] testbed with varying numbers of labeled target examples per category. We find that our method using soft label loss (with and without domain confusion) outperforms the baselines of training on source data alone or using a standard fine-tuning strategy to train with the source and target data. *Best viewed in color.*

CNNs fine-tuned using only source data using source and target labeled data. Contrary to the previous supervised adaptation experiment, our method significantly outperforms both baselines. We see that our full architecture, combining domain confusion with the soft label loss, performs the best overall and is able to operate in the regime of no labeled examples in the target (corresponding to the red line at point 0 on the x-axis). We find that the most benefit of our method arises when there are few labeled training examples per category in the target domain. As we increase the number of labeled examples in the target, the standard fine-tuning strategy begins to approach the performance of the adaptation approach. This indicates that direct joint source and target fine-tuning is a viable adaptation approach when you have a reasonable number of training examples per category. In comparison, fine-tuning on the target examples alone yields accuracies of 36.6 ± 0.6, 60.9 ± 0.5, and 67.7 ± 0.5 for the cases of 1, 3, and 5 labeled examples per category, respectively. All of these numbers underperform the source only model, indicating that adaptation is crucial in the setting of limited training data.

Finally, we note that our results are significantly higher than the 24.8% result reported

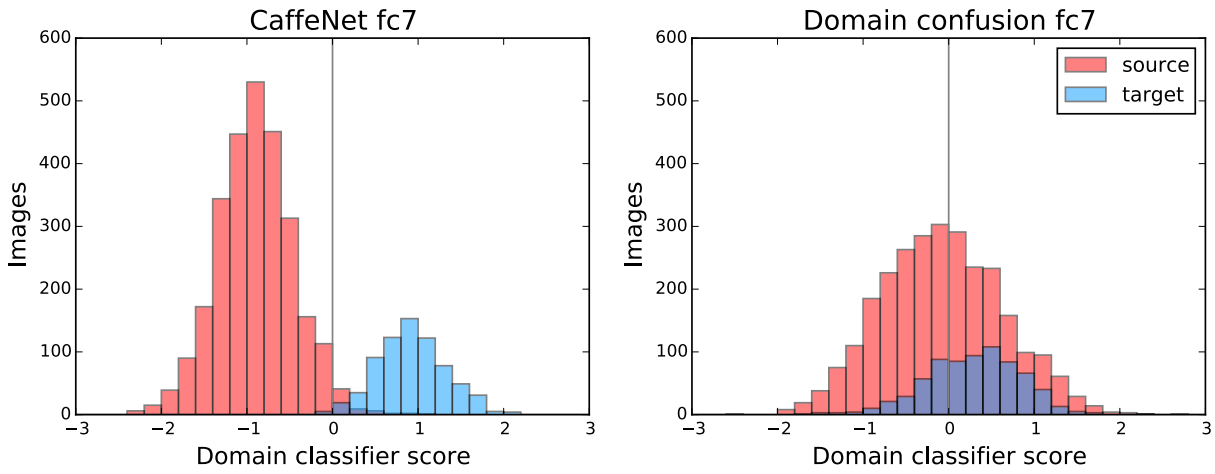


Figure 9.7: We compare the baseline CaffeNet representation to our representation learned with domain confusion by training a support vector machine to predict the domains of Amazon and Webcam images. For each representation, we plot a histogram of the classifier decision scores of the test images. In the baseline representation, the classifier is able to separate the two domains with 99% accuracy. In contrast, the representation learned with domain confusion is domain invariant, and the classifier can do no better than 56%.

in [174], despite the use of much less training data. This difference is explained by their use of SURF BoW features, indicating that CNN features are a much stronger feature for use in adaptation tasks.

9.5 Analysis

Our experimental results demonstrate that our method improves classification performance in a variety of domain adaptation settings. We now perform additional analysis on our method by confirming our claims that it exhibits domain invariance and transfers information across tasks.

Domain confusion enforces domain invariance

We begin by evaluating the effectiveness of domain confusion at learning a domain invariant representation. As previously explained, we consider a representation to be domain invariant if an optimal classifier has difficulty predicting which domain an image originates from. Thus, for our representation learned with a domain confusion loss, we expect a trained domain classifier to perform poorly.

We train two support vector machines (SVMs) to classify images into domains: one using the baseline CaffeNet fc7 representation, and the other using our fc7 learned with domain

confusion. These SVMs are trained using 160 images, 80 from Amazon and 80 from Webcam, then tested on the remaining images from those domains. We plot the classifier scores for each test image in Figure 9.7. It is obvious that the domain confusion representation is domain invariant, making it much harder to separate the two domains—the test accuracy on the domain confusion representation is only 56%, not much better than random. In contrast, on the baseline CaffeNet representation, the domain classifier achieves 99% test accuracy.

Soft labels for task transfer

We now examine the effect of soft labels in transferring information between categories. We consider the Amazon→Webcam shift from the semi-supervised adaptation experiment in the previous section. Recall that in this setting, we have access to target labeled data for only half of our categories. We use soft label information from the source domain to provide information about the held-out categories which lack labeled target examples. Figure 9.8 examines one target example from the held-out category *monitor*. No labeled target monitors were available during training; however, as shown in the upper right corner of Figure 9.8, the soft labels for *laptop computer* was present during training and assigns a relatively high weight to the *monitor* class. Soft label fine-tuning thus allows us to exploit the fact that these categories are similar. We see that the baseline model misclassifies this image as a *ring binder*, while our soft label model correctly assigns the *monitor* label.

9.6 Conclusion

We have presented a CNN architecture that effectively adapts to a new domain with limited or no labeled data per target category. We accomplish this through a novel CNN architecture which simultaneously optimizes for domain invariance, to facilitate domain transfer, while transferring task information between domains in the form of a cross entropy soft label loss. We demonstrate the ability of our architecture to improve adaptation performance in the *supervised* and *semi-supervised* settings by experimenting with two standard domain adaptation benchmark datasets. In the semi-supervised adaptation setting, we see an average relative improvement of 13% over the baselines on the four most challenging shifts in the Office dataset. Overall, our method can be easily implemented as an alternative fine-tuning strategy when limited or no labeled data is available per category in the target domain.

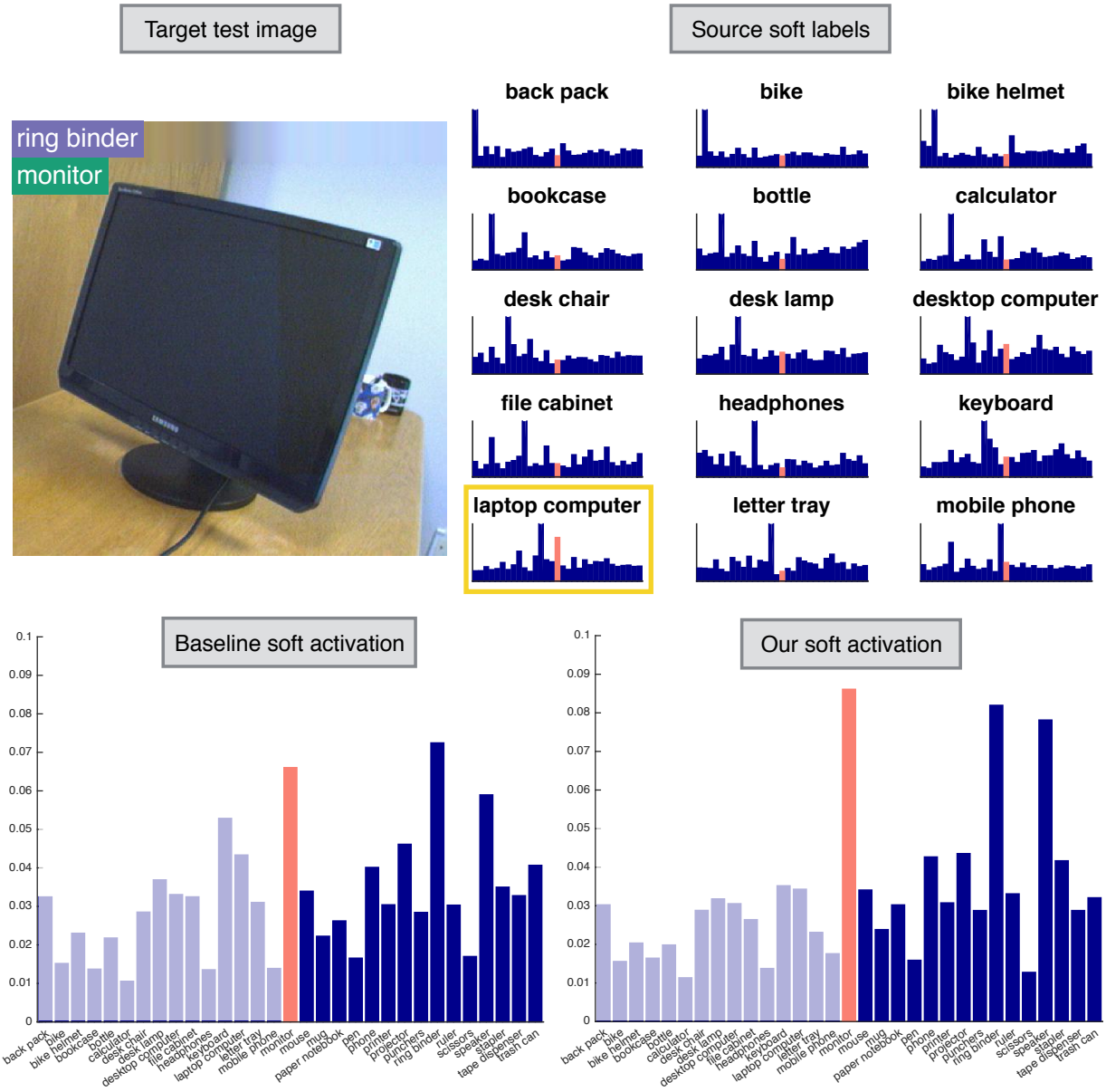


Figure 9.8: Our method (bottom turquoise label) correctly predicts the category of this image, whereas the baseline (top purple label) does not. The source per-category soft labels for the 15 categories with labeled target data are shown in the upper right corner, where the x-axis of the plot represents the 31 categories and the y-axis is the output probability. We highlight the index corresponding to the *monitor* category in red. As no labeled target data is available for the correct category, *monitor*, we find that in our method the related category of *laptop computer* (outlined with yellow box) transfers information to the monitor category. As a result, after training, our method places the highest weight on the correct category. Probability score per category for the baseline and our method are shown in the bottom left and right, respectively, training categories are opaque and correct test category is shown in red.

Part IV

Defining Visual Domains

Chapter 10

Visual Domains

Standard visual domain adaptation assumes a single source and target domain, where each domain is defined as a single dataset or image collection. Often this designation is useful since each dataset is collected within the vision community for the purpose of solving a single task and therefore the images of a single dataset often contain a similar bias.

The classic machine learning definition of domain adaptation says that a domain is defined as a distribution from which the sample data is drawn and a shift between domains corresponds to a difference, or distance, between the source and target distribution. In this part of the thesis we challenge the notion of a visual dataset comprising a single domain arguing instead that many datasets may be composed of multiple domains. We demonstrate two scenarios where this is particularly true.

In Chapter 11, we consider a collection of internet images and demonstrate that through a hierarchical clustering approach we are able to distinguish multiple latent domains within the single dataset. Furthermore, we present a multiple-source domain adaptation approach to utilize all source domain information to produce a stronger adaptation model for the target classification. We additionally present a novel theoretical analysis of multiple-source domain adaptation in Chapter 12.

Finally, in Chapter 13, we study the problem of a continuously evolving target domain. For many applications, a recognition model is deployed at some fixed time and then continues to be used over time. However, for many situations, such as the outdoor surveillance setup we study, the data varies significantly over time. We present a continuous manifold adaptation algorithm which can be used for unsupervised adaptation to allow our initial source model to be modified as the target domain evolves.

Chapter 11

Discovering Latent Domains

11.1 Introduction

Despite efforts to the contrary, most image datasets exhibit a clear *dataset bias*: supervised learning on a particular dataset nearly always leads to a significant loss in accuracy when the models are tested in a new domain [151, 175]. *Domain Adaptation* methods have been proposed as a solution to dataset bias and are becoming increasingly popular in computer vision. Especially attractive are recent weakly-supervised methods that learn to transform features between domains based on partially labeled data [151, 104, 64].

A major limitation of these methods is the assumption that the domain/dataset label is provided for each training image. However, in practice, one often has access to large amounts of object labeled data composed of multiple *unknown domains*. For example, images found on the web can be thought of as a collection of many hidden domains. As shown in Figure 11.1, image search results for "person" and "bicycle" consist of several *types*, such as close-up photos of a face, photos of an entire human figure, group shots, line drawings of a person, etc. Existing methods have not addressed the problem of separating such data into latent domains for the purpose of domain adaptation.

Even when the separation into source domains is known, the above methods are limited to a single adaptive transform between source and target data, computed by either simply pooling [151, 104] or averaging [64] multiple sources into one source domain. Learning a single transform may be sub-optimal in approximating multiple domain shifts. Consider a simple example: two source image datasets, one with *low-resolution*, *color* images and one with *high-resolution*, *grayscale* images. When mapping images from a novel *high-resolution*, *color* domain to each of these source domains, we would like a transform that either discounts the high-resolution details or the color, but not both. Rather than force a single transform to perform both mappings, a richer model with multiple separate transforms is desirable.

The contribution of this paper is two-fold: First, we propose a method that discovers latent domain labels in data which has heterogeneous but unknown domain structure. We use a probabilistic framework to derive a hierarchical constrained assignment algorithm that

Figure 11.1: Training images for object recognition may contain several unknown domains, such as the line drawings, close-up photos and far-away shots returned by web image search for *person* and *bicycle*.

separates heterogeneous training data into latent clusters, or domains. Our second contribution is an extension of the feature-transform method in [104] to multi-domain adaptation that uses source domain labels to define a mixture-transform model. Tests on standard datasets with known domain labels confirm that our method is more accurate at discovering domain structure than baseline clustering methods, and that our transform mixture model outperforms a single transform approach. When domain labels are unknown, we evaluate the end-to-end recognition performance with no adaptation baseline, baseline adaptation, and transform-mixture adaptation with discovered domain labels (our method). Our experiments on category data from the *Bing* image search dataset confirm improved classification performance using the inferred domains.

11.2 Related Work

Domain adaptation aims to compensate for covariate shift, i.e. a change in the feature distribution from training to test time, which is a version of the more general dataset shift problem [144].

Recently, several authors have developed approaches for vision tasks, and the field is becoming increasingly aware of existing dataset bias [175]. These methods can be roughly categorized as classifier adaptation approaches and feature-transform approaches. The former approach adapts the parameters of each per-category binary classifier, typically a support vector machine, and includes methods like a weighted combination of source and target SVMs; transductive SVMs applied to adaptation in [16]; Adaptive SVM [189], where the parameters are adapted by adding a perturbation function; Domain Transfer SVM [46], which learns a target decision function while reducing the mismatch in the domain distributions; and a related method [47] which utilizes adaptive multiple kernel learning to learn a kernel function based on multiple base kernels. Classifier adaptation approaches are typically supervised, requiring labeled examples for each category in both source and target.

On the other hand, feature transform approaches map or translate the input features directly between domains, and then apply a classifier [151, 104, 64, 50, 30]. The advantage

of these approaches over the classifier-based ones above is that they are able to transfer the adaptive transformation to novel problems, and even deal with new feature types and dimensionalities. For example, the asymmetric nonlinear transform method [104] learns a domain-independent similarity function between arbitrary feature sets based on class constraints. In this paper we focus on transform learning, as it can be scaled up to many novel object categories and heterogeneous features. Other work on feature-translation has included translating features between camera views to transfer activity models [50], and translating user preferences between text and image domains [30].

Some of the existing methods can accommodate multiple source domains, if they are known. The feature replication method [32] can be easily extended to multiple sources. In methods based on combinations of SVM classifiers, this is done primarily by weighting the classifiers learned on each source domain, either equally, or as in [45], using the maximum mean discrepancy (MMD) criterion, which measures the distance between distributions. The A-MKL [47] learns an optimal weighted combination of the pre-learned classifiers, however, in the evaluation the classifiers came from the same source domain. The unsupervised approach based on Grassman manifolds presented by Gopalan et al. [64] supports multiple sources by first computing the Karcher mean of the sources and then running the single source version of their algorithm.

None of the existing methods address the case of domain adaptation with unknown domain labels, the main focus of our paper. We also present the first multi-domain version of the asymmetric nonlinear transform [104], which so far has been limited to the single domain case.

Our domain discovery method is based on a constrained clustering method, a topic of active research for several years. The most related work to our approach is based on constrained k-means. The method of Wagstaff et al. [181] is the earliest constrained clustering method based on k-means; their algorithm greedily assigns points to the closest cluster which does not violate any constraints. Improvements to this basic algorithm were considered in [9, 105]. A variant of constrained k-means was also used recently with soft constraints to learn more discriminative codebooks [173]. In general, the constraints in these approaches are incorporated as additional penalty terms to the k-means objective function (and therefore are not guaranteed to be satisfied), and algorithms are developed to minimize the penalized k-means objective. In contrast to existing work on constrained k-means, our algorithm is guaranteed to find a clustering that satisfies the constraints and provably converges locally under certain assumptions. An overview of recent semi-supervised clustering techniques, including techniques not based on k-means, can be found in [10].

11.3 Single Transform Domain Adaptation

We review the single-transform method of Kulis et al. [104]. Suppose we have a source domain $\mathcal{X}$ containing observations $\mathbf{x}_1, ..., \mathbf{x}_{n_\mathcal{X}}$. Similarly, let $\mathcal{V}$ be the target domain, containing observations $\mathbf{v}_1, ..., \mathbf{v}_{n_\mathcal{V}}$. Suppose we are also given a set of labels $y_1, ..., y_{m_\mathcal{X}}$ for the

source observations and a partial set of labels $g_1, ..., g_{m_\mathcal{V}}$ for the target observations, with $m_\mathcal{V} < m_\mathcal{X}$ and $l \in \{1, ..., K\}$, where K is the number of categories. The goal of domain transform learning is to estimate a semantic similarity function $sim(\mathbf{x}, \mathbf{v})$ which outputs high similarity values for pairs of source examples $\mathbf{x}$ and novel target examples $\mathbf{v}$ if they have the same label, and low similarity values if they have different labels.

Following [104], consider a similarity function between source and target data parametrized by a matrix $\boldsymbol{W}$, i.e., $sim_{\boldsymbol{W}}(\mathbf{x}, \mathbf{v}) = \phi_\mathcal{X}(\mathbf{x})^T \boldsymbol{W} \phi_\mathcal{V}(\mathbf{v})$, where $\phi_\mathcal{X}$ and $\phi_\mathcal{V}$ map examples from the source and target, respectively, into kernel space. In general, mapping to kernel space makes it possible to learn non-linear similarity mappings in the input space.

The transform $\boldsymbol{W}$ can be learned via a regularized optimization problem with loss functions that depend on $sim_{\boldsymbol{W}}$; that is, by optimizing for a matrix $\boldsymbol{W}$ which minimizes $r(\boldsymbol{W}) + \gamma \sum_{i=1}^{C} c_i(\phi_\mathcal{X}(\mathbf{X})^T \boldsymbol{W} \phi_\mathcal{V}(\mathbf{V}))$, where r is a matrix regularizer, C is the number of constraints, $\phi_\mathcal{X}(\mathbf{X})$ is the matrix of source data mapped to kernel space, $\phi_\mathcal{V}(\mathbf{V})$ is the matrix of target data mapped to kernel space, and c_i are loss functions over the matrix $\phi_\mathcal{X}(\mathbf{X})^T \boldsymbol{W} \phi_\mathcal{V}(\mathbf{V})$. Though the model can incorporate various constraints/losses, it is customary to focus on constraints that penalize small values of $sim_{\boldsymbol{W}}(\mathbf{x}, \mathbf{v})$ when $\mathbf{x}$ and $\mathbf{v}$ share a class label, and penalize large values of $sim_{\boldsymbol{W}}(\mathbf{x}, \mathbf{v})$ when they have different labels. For a particular class of regularizers r, this problem may be efficiently optimized in kernel space; see [104].

11.4 Multiple Transform Domain Adaptation

We first extend the single source feature transform method [104] to a multi-source algorithm, considering the case of known domain labels, and then in Section 11.5 we present our main contribution: a method for discovering unknown domain labels.

Transform-based domain adaptation captures category-independent domain shift information, which can be generalized to a held-out category for which no labels are available in a new domain. The single-transform method described in Section 11.3 is sub-optimal when multiple domain shifts are present. We remedy this by constructing a domain transform mixture model as follows.

Suppose we have a set of domains $\mathcal{X}_1, ..., \mathcal{X}_S$ containing observations $\mathbf{x}_1, ..., \mathbf{x}_n$ and labels $y_1, ..., y_n\}$. Let $\boldsymbol{a} = \{a_1, ..., a_n\}$ specify the domain of each observation, such that $a_i \in \{1, ..., S\}$.

We start by using the single source feature transform method to learn a transformation $\boldsymbol{W}_k$ for each source $k \in \{1, \ldots, S\}$. Each of the S different object category classifiers outputs a multi-class probability for a given test point, $\mathbf{v}$. We denote the output probability over classes, c, of the classifier learned for the kth domain as, $p(c|d = k, \mathbf{v})$.

We classify a novel test point from a target domain by considering the domain as a latent

variable that is marginalized out as follows:

$$\text{label}(\mathbf{v}) = arg \max_{c \in \{1:K\}} p(c|\mathbf{v}) \quad (11.1)$$

$$= arg \max_{c \in \{1:K\}} \sum_{k=1}^{S} p(d=k|\mathbf{v}) p(c|d=k, \mathbf{v}) \quad (11.2)$$

Where $p(d|\mathbf{v})$ is the output of a domain label classifier and $p(c|d, \mathbf{v})$ is the output of a domain specific object model based on the learned transforms. For example, if the domain specific classifier is (kernelized) nearest neighbors, as in our experiments, and we let $\mathbf{x}_k^*(\mathbf{v})$ be the most similar point in domain k to test point $\mathbf{v}$, then our domain specific object category probability can be expressed as:

$$p(c|d=k, \mathbf{v}) = \frac{\phi_{\mathcal{X}}(\mathbf{x}_k^*(\mathbf{v}))^T \boldsymbol{W}_j \phi_{\mathcal{V}}(\mathbf{v})}{\sum_{k'=1}^{S} \phi_{\mathcal{X}}(\mathbf{x}_{k'}^*(\mathbf{v}))^T \boldsymbol{W}_j \phi_{\mathcal{V}}(\mathbf{v})} \quad (11.3)$$

Finally, to obtain $p(d=k|\mathbf{v})$ we train an SVM classifier with probabilistic outputs, using known domain labels and source training data. In summary, category classification of a test point $\mathbf{v}$ amounts to a weighted sum of the probability of a particular category given that the point is from a particular domain, where the weights are the probability that the test point belongs to each domain.

11.5 Domain Clustering

For datasets which do not have domain labels, we must infer domain assignments $\hat{\boldsymbol{a}}$ that best approximate the true assignments, $\boldsymbol{a}$. This task is difficult because, in many cases, the data is naturally separated according to semantic categories. That is, a standard clustering method such as k-means or EM for mixtures of Gaussians would tend to return clusters based on the semantic category labels, which is clearly undesirable.

High-level Description: We propose a two-stage approach for domain discovery (See Figure 11.2 for illustration of algorithm). The idea is as follows: if there are K semantic categories and S domains in the data, we will group the data points $\mathbf{x}_1, ..., \mathbf{x}_n$ into $J = K \cdot S$ *local clusters*—intuitively, each local cluster will contain data points from one domain and one semantic category (step B-C in Figure 11.2). In the second stage of clustering (*domain clustering*), we will cluster the means of the local clusters, in order to find domains (step C-D in Figure 11.2). The two stages are iterated to convergence. To make this approach consistent with our goals, we add two crucial constraints to the clustering procedure: a) we require that each local cluster only contains data points from a single category, and b) we constrain each domain cluster to contain only one local cluster from each object category.

Formal Definition: Let us now specify this model more formally. Define a local latent

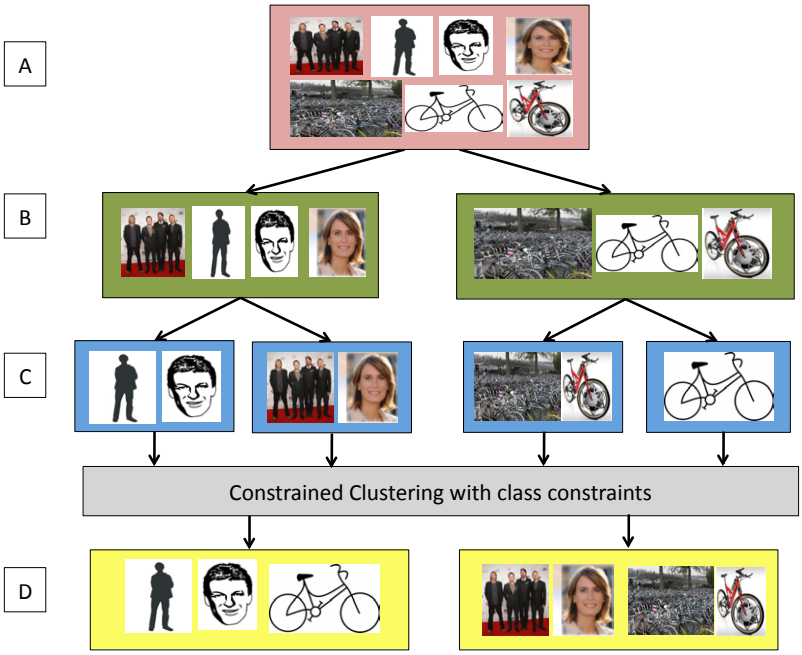


Figure 11.2: An illustration of our approach for discovering latent domains: Given input images with unknown domains types (A), we separate examples according to their semantic category labels (B). Then, we cluster the data from each semantic category group to produce local clusters (C), and, finally, use do-not-link constraints between clusters of same category to produce the domain specific clusters (D). This example shows separation of *person* and *bicycle* search results into two domains of line drawing and natural images. Our algorithm iterates between steps (B-D) until convergence.

variable $z_{ij}^L \in \{0,1\}$ to indicate whether observation $\mathbf{x}_i$ should be assigned to local cluster j. Let the domain latent variable $z_{jk}^G \in \{0,1\}$ indicate whether local cluster j is assigned to domain k. Further, let $\boldsymbol{\mu}_j$ be the mean of the observations in local cluster j and $\boldsymbol{m}_k$ be the mean of domain cluster k. Finally, let the mixing weights for the global clusters be given by π_k^G, for $k = 1, ..., S$, and let the mixing weights for the local clusters be given by π_j^L for $j = 1, ..., J$. See Figure 11.3 for a graphical depiction of our model. Note that, unlike a standard Gaussian mixture model, each cluster mean $\boldsymbol{\mu}_j$ is itself a data point within a mixture model. For simplicity, we are assuming that all clusters have a global covariance of σI—one could easily extend the model to learn the covariances. The conditional probabilities in our

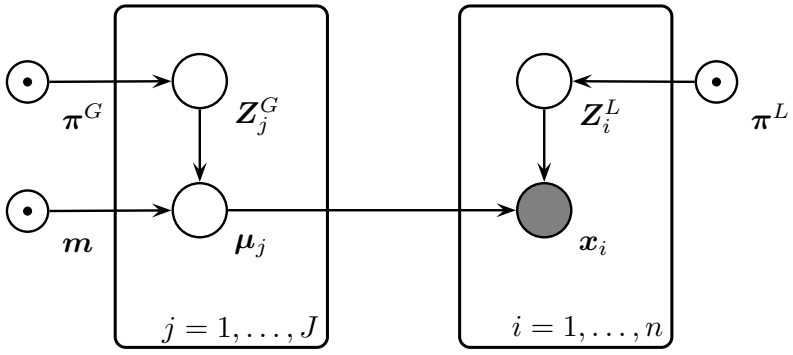


Figure 11.3: Graphical model for domain discovery. The data points $\mathbf{x}_i$ are generated from category specific local mixtures, whose means are assumed to be drawn from a latent global (*domain*) mixture.

graphical model are defined as:

$$p(\boldsymbol{z}_i^L \mid \boldsymbol{\pi}^L) = \prod_{j=1}^{J} (\pi_j^L)^{z_{ij}^L} \qquad p(\boldsymbol{z}_j^G \mid \boldsymbol{\pi}^G) = \prod_{k=1}^{S} (\pi_k^G)^{z_{jk}^G} \tag{11.4}$$

$$p(\boldsymbol{x}_i \mid \boldsymbol{z}_i^L, \boldsymbol{\mu}) = \prod_{j=1}^{J} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \sigma I)^{z_{ij}^L} \tag{11.5}$$

$$p(\boldsymbol{\mu}_j \mid \boldsymbol{z}_j^G, \boldsymbol{m}) = \prod_{k=1}^{S} \mathcal{N}(\boldsymbol{\mu}_j \mid \boldsymbol{m}_k, \sigma I)^{z_{jk}^G} \tag{11.6}$$

We may think of this model in a generative manner as follows: for each local cluster $j = 1, ..., J = S \cdot K$, we determine its underlying domain via the $\boldsymbol{\pi}^G$ mixing weights. Given this domain, we generate $\boldsymbol{\mu}_j$, the mean for the local cluster. Then, to generate the data points $\boldsymbol{x}_i$, we first determine which local cluster j the data point belongs to using $\boldsymbol{\pi}^L$, and then we generate the point from the corresponding $\boldsymbol{\mu}_j$.

At this point, we still need to add the constraints discussed above, namely that the local clusters only contain data points from a single category and that domain clusters contain only a single local cluster from each category. Such constraints can be difficult to enforce in a probabilistic model, but are considerably simpler in a hard clustering model. For instance, in semi-supervised clustering there is ample literature on adding constraints to the k-means objective via constraints or penalties [181, 105]. Thus, we will utilize a standard asymptotic argument on our probabilistic model to create a corresponding hard clustering model. In particular, if one takes a mixture of Gaussian model with fixed σI covariances across clusters, and lets $\sigma \to 0$, the expected log likelihood approaches the k-means objective, and the EM

algorithm approaches the k-means algorithm. In an analogous manner, we will consider the log-likelihood of our model. Let $\boldsymbol{Z}^L = \{\boldsymbol{z}_i^L\}$, $\boldsymbol{Z}^G = \{\boldsymbol{z}_j^G\}$, and $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$.

$$\ln \left[p(\boldsymbol{Z}^G, \boldsymbol{Z}^L, X \mid \boldsymbol{m}, \boldsymbol{\mu})\right] = \tag{11.7}$$
$$\sum_{i=1}^{n}\sum_{j=1}^{J}\sum_{k=1}^{S}\left(z_{jk}^G(\ln \pi_k^G + \ln \mathcal{N}(\boldsymbol{\mu}_j \mid \boldsymbol{m}_k, \sigma I)) + z_{ij}^L(\ln \pi_j^L + \ln \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \sigma I))\right)$$

If we have no prior knowledge about the domains, then we assume the mixing weights π^G and π^L to be uniform. Then to create a hard clustering problem we let σ tend to 0 and taking expectations with respect to $\boldsymbol{Z}^G$ and $\boldsymbol{Z}^L$, we obtain the following hard clustering objective:

$$\min_{\boldsymbol{Z}^G, \boldsymbol{Z}^L, \boldsymbol{\mu}, \boldsymbol{m}} \quad \sum_{i=1}^{n}\sum_{j=1}^{J}\boldsymbol{Z}_{ij}^L(x_i - \mu_j)^2 + \sum_{j=1}^{J}\sum_{k=1}^{S}\boldsymbol{Z}_{jk}^G(\mu_j - \boldsymbol{m}_k)^2 \tag{11.8}$$

$$\text{subject to:} \quad \forall j,k: \ \boldsymbol{Z}_{jk}^G \in \{0,1\}, \quad \forall i,j: \ \boldsymbol{Z}_{ij}^L \in \{0,1\} \tag{11.9}$$

$$\forall j: \ \sum_{k=1}^{S}\boldsymbol{Z}_{jk}^G = 1, \quad \forall i: \ \sum_{j=1}^{J}\boldsymbol{Z}_{ij}^L = 1 \tag{11.10}$$

$$\tag{11.11}$$

The constraints above are standard hard assignment constraints, saying that every observation must be assigned to one local cluster and every local cluster must be assigned to only one global cluster. Now that we have transformed the clustering problem into a hard clustering model, we can easily add constraints on the $\boldsymbol{Z}^G$ and $\boldsymbol{Z}^L$ assignments, leading to our final model.

$$\min_{\boldsymbol{Z}^G, \boldsymbol{Z}^L, \boldsymbol{\mu}, \boldsymbol{m}} \quad \sum_{i=1}^{n}\sum_{j=1}^{J}\boldsymbol{Z}_{ij}^L(x_i - \mu_j)^2 + \sum_{j=1}^{J}\sum_{k=1}^{S}\boldsymbol{Z}_{jk}^G(\mu_j - \boldsymbol{m}_k)^2 \tag{11.12}$$

$$\text{subject to:} \quad \forall j,k: \ \boldsymbol{Z}_{jk}^G \in \{0,1\}, \quad \forall i,j: \ \boldsymbol{Z}_{ij}^L \in \{0,1\} \tag{11.13}$$

$$\forall j: \ \sum_{k=1}^{S}\boldsymbol{Z}_{jk}^G = 1, \quad \forall i: \ \sum_{j=1}^{J}\boldsymbol{Z}_{ij}^L = 1 \tag{11.14}$$

$$\forall j: \ \sum_{i=1}^{n}\delta(l_i \neq l_j)\boldsymbol{Z}_{ij}^L = 0 \tag{11.15}$$

$$\forall k: \ \sum_{c=1}^{K}\sum_{j=1}^{J}\delta(\text{label}(j) = c)\boldsymbol{Z}_{jk}^G = 1 \tag{11.16}$$

Equation (11.15) says that all observations assigned to a single local cluster j must contain the same object category label. Finally, we add the constraint in Equation (11.16), which

says that each global cluster should only contain one local cluster from each object category. Here, we define label(j) to mean the object category label of the observations assigned to the local cluster j. Together the last two constraints restrict the feasible global clustering solutions to be the solutions that contain observations from every object category while placing observations together that are close according to a Euclidean distance.

To solve this optimization problem we formulate an EM style iteration algorithm. We first initialize $\boldsymbol{Z}^G, \mu, \boldsymbol{m}$ and then perform the following updates until convergence (or until the objective doesn't change more than some threshold between iterations):

$\boldsymbol{Z}^L_{ij}$: For each observation $\boldsymbol{x}_i$, set $\boldsymbol{Z}^L_{ij} = 1$ for the particular j that minimizes $(\boldsymbol{x}_i - \mu_j)^2$.

μ_j: For each local cluster j, set $\mu_j = \frac{\sum_i \boldsymbol{Z}^L_{ij}\boldsymbol{x}_i + \sum_k \boldsymbol{Z}^G_{jk}\boldsymbol{m}_k}{\sum_i \boldsymbol{Z}^L_{ij} + \sum_k \boldsymbol{Z}^G_{jk}}$

$\boldsymbol{Z}^G_{jk}$: For each local cluster j, set $\boldsymbol{Z}^G_{jk} = 1$ for the particular global cluster that minimizes $\boldsymbol{Z}^G_{jk}(\mu_j - \boldsymbol{m}_k)^2$ while satisfying Equation (11.16). For the case where the number of latent domains is assumed small, we can try all possible assignments of local to global clusters at this stage.

$\boldsymbol{m}_k$: For each global cluster k, set $\boldsymbol{m}_k = \frac{\sum_j \boldsymbol{Z}^G_{jk}\mu_j}{\sum_j \boldsymbol{Z}^G_{jk}}$

Note, these updates correspond to constrained versions of the EM updates from the probabilistic model as $\sigma \to 0$. In general adding constraints to EM updates can cause the objective to diverge. However, for our problem, when the number of domains, S, is small (which is a practical assumption for many domain adaptation tasks) these updates can be shown to provably converge locally (see supplemental material). In Section 11.6 we show experimentally that even for large heterogeneous data sources learning to separate into a small number of latent domains is sufficient in practice.

We evaluate absolute domain discovery performance compared against a constrained hard HDP method [106] and a standard unconstrained k-means. The hard HDP seeks to minimize the distance between data points and their local mean plus data points to their assigned global mean, whereas we introduce an intermediate latent variable for the local cluster means. That, combined with the fact that constrained hard HDP can create new global clusters, causes the constrained hard HDP to degenerate to the case where each global cluster contains only one local cluster. In Section 11.6 we present clustering accuracy experiments that show our method does in fact provide significant benefit over the hard HDP method and a standard k-means baseline.

11.6 Experiments

We present the following three experiments. First, we evaluate our multi-domain mixture model extension of [104] described in Section 11.4 for the case where the domain labels are known. We compare this to using the single-transform method in [104]. Next, we evaluate

the ability of our constrained clustering method to recover domains by mixing datasets and omitting the dataset labels, and compare to two baseline clustering methods. Finally, we evaluate the end-to-end recognition performance on a dataset with unknown domains, compared against methods with no adaptation, baseline adaptation, and our multiple transform adaptation with discovered domain labels.

Datasets: For our experiments we used two different data sets. First we used the *Office* data set created by [151] specifically for object domain adaptation, which we modified to artificially introduce more domains. We use the code provided by the authors to extract features and to implement the single-domain asymmetric transform method. Second, we evaluate on a subset of the *Bing-Caltech256* data set created by [16], using the authors' original image features. The *Bing* dataset is comprised of search results from object label keywords and therefore has many sub-domains without any explicit labels.

The *Office* dataset contains three domains with 31 object categories: Amazon (a), which contains images from amazon.com; Digital SLR (d), and Webcam (w). Webcam and Digital SLR contain images of the same 5 instances for each object category, but vary in pose, lighting, and camera. In addition to these three domains, we also created two more domains which are artificially modified versions of two the the original domains: Webcam-blur (Wb), which contains the blurred version of all webcam data images (using a Gaussian filter of width 5), and Amazon-rotflip (Ar), which contains the result of rotating all amazon images by 10-20 degrees and then creating their mirror image.

Object Classification using *Office*: For our first object classification experiment, we follow the experimental procedure for semi-supervised learning described in [151, 64]: we train the transforms on categories 1-15, with 10 instances per category in each source domain and 5 instances per category in the target domain, and test them using the classifier described in Section 11.4 on the rest of the categories.

The experiment measures the accuracy of predicting the labels of target data points with no examples of the correct label within the target domain, representing the transform's ability to capture general shifts rather than simply the source shift's effect on a particular object category.

We compared our method to the single-transform method of [104] and the semi-supervised multi-domain version of [64]. The γ parameter of ours and [104] was optimized for the single transform baseline in all experiments. We ran code provided by the authors of [64]; we experimented with over 1000 combinations of the four parameter settings within the ranges specified in the code from the author's website and the paper, but were unable to produce resulting accuracies above the standard KNN with no adaptation baseline for any of the domain shifts presented, and so we omit these results from the following figure.

Figure 11.4 shows the absolute improvement in accuracy of the supervised multiple-domain method and a single transform method over the KNN classifier using Euclidean distance (i.e., no adaptation) for various domain shifts, averaged over 10 train/test splits for each shift. Overall, the average accuracy gained over KNN was 3.27 for the single source baseline [104] and 5.45 for our domain-supervised method. The average KNN classification accuracy was 28.9, which means these absolute improvements correspond to an 11.3% and

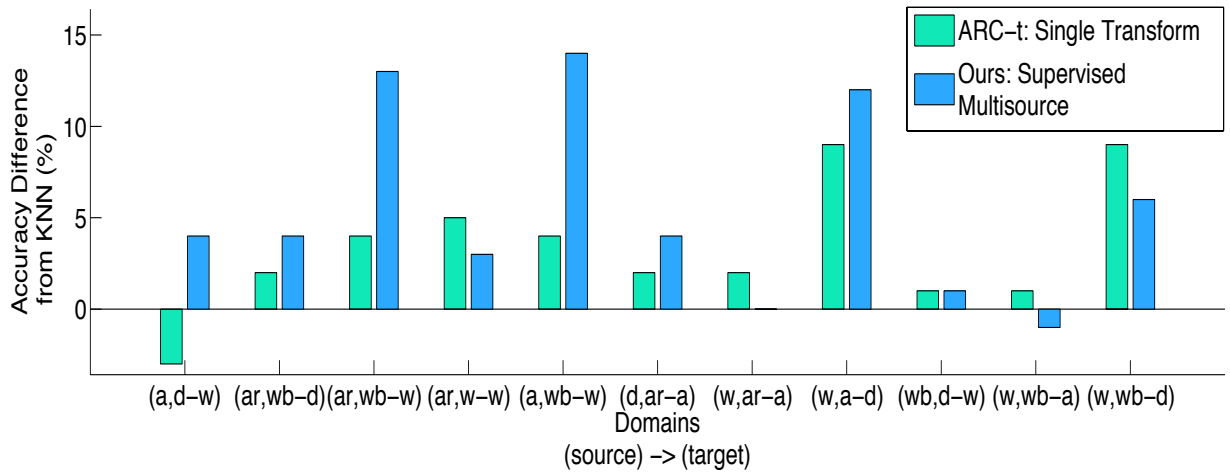


Figure 11.4: Classification Accuracy experiment on *Office* dataset with known domain labels. The plot shows the average accuracy improvement of each method over a baseline KNN. Each cluster of bars is the results for a different domain shift, where the domains are indicated on the x-axis. For each of these cases the parameter, γ was set to be the value that maximized the single transform method baseline. The average accuracy gained over KNN across all datasets was 3.27 for the single source baseline [104] and 5.45 for our multisource domain-supervised method. For some datasets our method outperforms [104] significantly, achieving up to a 18.9% relative improvement over KNN.

18.9% relative improvement over KNN for each method respectively.

We notice that shifts where the supervised method achieves the most gains (up to 48% relative gain in accuracy) have higher separability of the source domains via clustering reported below (in Figure 11.6.)

Clustering: We begin with an analysis of our clustering algorithm on all pairs of the 5 domains in the *Office* dataset, for which we know the ground truth source labels. Figure 11.6 plots the accuracy improvement over chance between the ground truth domain labels and the cluster labels learned using our constrained clustering algorithm. The results are averaged over 10 runs with different data splits for each run. As baselines, we implemented standard, unconstrained, k-means, as well as, a hard HDP [106] with the same do-not-link class based constraints presented in Section 11.5.

Our clustering method outperforms these baselines and in general has high accuracy when compared with the ground truth domain labels. The one exception is when clustering the pair (Amazon-rotflip, Amazon). In this case, the domains represented in our feature space are very similar and so there is no adequate separation for the clustering algorithm to find. The last bar in Figure 11.6 shows the mean clustering accuracy across all domain shifts and shows the significant improvement of using our method to cluster into domains.

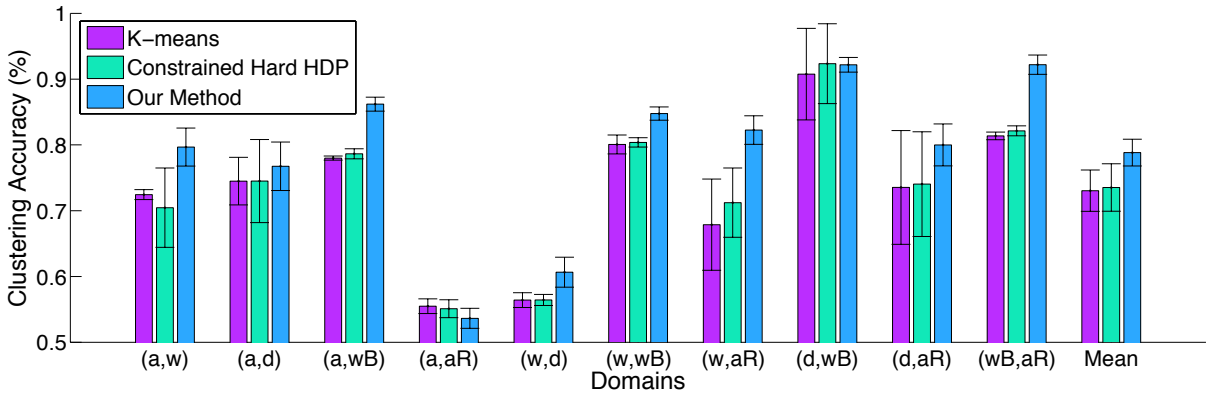


Figure 11.5: Clustering quality, measured in terms of accuracy of cluster labels. Each cluster of bars represents a mixed pair of domains from the *Office* dataset, see text. The results are averaged over multiple data splits. Our method significantly outperforms the baselines in 7 out of the 10 datasets.

Object Classification using *Bing-Caltech256*: We show the results of running our end-to-end algorithm on the Bing web search and Caltech256 dataset created by [16]. Here we assume the Bing images contain sub-domains. We use the first 30 categories from each of Bing and Caltech, and assume that all points in Bing have strict class labels (in reality, the results of web search are only weakly labeled.) There are no domain labels for the *Bing* source dataset so we present qualitative clustering results in Figures 11.6(a-c). The clusters shown represent intuitive domain distinctions, justifying out method: 11.6a cartoon/drawing/synthetic images with limited illumination, 11.6b product images with simple backgrounds and some illumination, 11.6c natural or cluttered scenes.

After clustering, we train the transforms on the first 15 categories in the Caltech256 domain and use categories 16 to 30 for testing, again testing the prediction accuracy of transformed data points into new object categories. We set $S = 2$ and use the SVM-based domain classifier described in Section 11.4 for the mixture model.

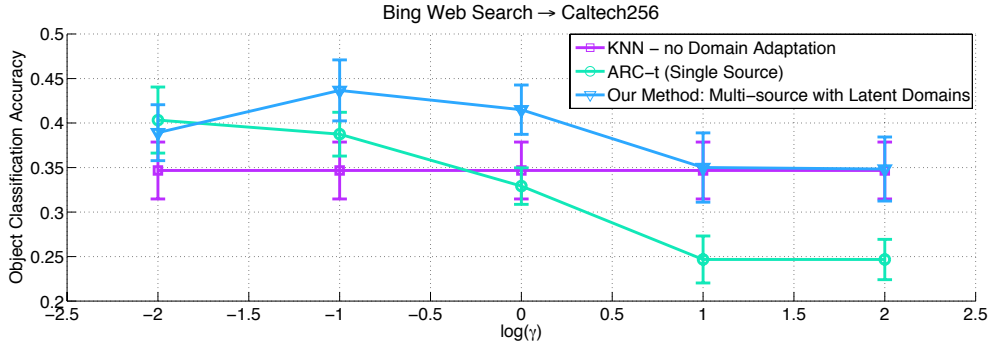
Figure 11.6 plots the results of running our algorithm with 20 instances per category in the source domain and 10 instances per category in the target domain. Ideally we would cross-validate the learning parameter γ, but due to lack of time we report results over varying γ on the test set. Our method obtains better results for a larger range of γ. The best performance obtained by the baseline is around 40% accuracy, while our method obtains 44% accuracy. This shows the advantage of using our method instead of pooling the data to learn a single transform, and that it is possible to learn domain clusters and to use them to effectively produce a better-adapted object classifier.

We presented a novel constrained clustering algorithm to distinguish unique domains from a large heterogeneous domain. Additionally, we proposed a simple multisource extension to the nonlinear transform-based ARC-t method of [151, 104]. Our algorithm provides the

(a) Cluster 1

(b) Cluster 2

(c) Cluster 3



(d) Classification Result

Figure 11.6: Results on the *Bing-Caltech256* dataset. (a-c) show example images from clustering the *Bing* dataset into three clusters. For each domain cluster the the three images closest to the domain cluster mean are shown from the categories baseball bat, basketball hoop, beer mug, bowling pin, and boxing glove are shown. The clusters represent intuitive domain definitions: (a) cartoon/drawings/synthetic images with limited illumination, (b) product images with simple backgrounds and some illumination, (c) natural or cluttered scenes. (d) Shows average accuracy versus the log of the learning parameter γ. Our unsupervised multi-domain method outperforms the use of a single learned transform, which falls below the KNN baseline as more weight is put on the constraints in the optimization problem.

ability to separate heterogeneous source data and learn multiple transforms, which creates a more accurate mapping of the source data onto the target domain than a single transform map. Our experiments illustrate that the multiple transformations can be effectively applied to new object categories at test time.

Thus far our experiments have focused on the case of multiple domains in the source and only one domain in the target. In future work, we plan to test our algorithm with multiple target domains. This should be a direct extension by clustering the target domain using the method described in Section 11.5.

Chapter 12

Multiple Source Domain Adaptation

12.1 Introduction

In many modern applications, often the learner has access to information about several source domains, including accurate predictors possibly trained and made available by others, but no direct information about a target domain for which one wishes to achieve a good performance. The target domain can typically be viewed as a combination of the source domains, that is a mixture of their joint distributions, or it may be close to such mixtures.

Such problems arise commonly in speech recognition where a different acoustic model is available for different domains, groups of speakers in this case, and where the problem consists of deriving an accurate acoustic model for a broader population that may be viewed as a mixture of the source groups [118]. Similar problems appear broadly in visual recognition where multiple image databases exist each with its own bias and labeled categories [175], but the target application may contain some combination of image types and any category may need to be recognized. For example, there are often multiple classifiers available, each trained using different source product images and large manually curated datasets, i.e. ImageNet [35], but the target domain may contain a different image population. Many have considered the case of transferring between a single source and known target domain either through unsupervised adaptation (no labels in the target) techniques [63, 124, 57, 177] or supervised (some labels in the target) [151, 189, 83, 59]. These problems also arise in sentiment analysis where accurate predictors may be available for sub-domains such as TVs, laptops and CD players for which labeled training data was at the learner's disposal, but not for the more general category of electronics, which can be modeled as a mixture of the sub-domains [18, 43].

How can the learner combine relatively accurate predictors available for source domains to derive an accurate predictor for a target domain? This is known as the *multiple-source adaption problem* first formalized and analyzed theoretically by [127, 128]. Note that, in the most general setting, even unlabeled information may not be available about the target domain, though one can expect it to be a mixture of the source distributions. Thus, the

problem is also closely related to domain generalization [137, 133, 187], where knowledge from an arbitrary number of related domains is combined to perform well on a previously unseen domain.

The main objective of this work is to design an efficient algorithm for combining multiple source models to derive an accurate model for *any* target domain that is a mixture of the source domains. Note that no algorithm was previously given by [127] for this problem. The algorithmic problem was in fact left as a difficult question since it required finding the solution of a Brouwer fixed-point problem.

Here, we build on the previous work by [127] but extend it to develop a general theoretical analysis of multiple-source adaptation for the stochastic scenario, that is the scenario where there is a distribution over the joint feature and label space, $\mathcal{X} \times \mathcal{Y}$, as opposed to one where a unique target labeling function is assumed. This generalization is needed to cover the realistic cases in applications. In particular, we apply our analysis to the standard loss used for training convolutional neural networks (CNNs), cross-entropy.

Our analysis shows, as in the special case analyzed by [127, 128], that there exists a remarkable predictor that admits a small expected loss with respect to *any* mixture distribution. We further extend this result to an arbitrary distribution with small Rényi divergence with respect to the family of mixtures. We also extend it to the case where, instead of having access to the ideal distributions, only estimate distributions are used for deriving that hypothesis.

Next, we give a new formulation of the problem of finding that robust predictor. We show that, in the important case of the cross-entropy loss, the problem can be cast as a DC-programming problem and we present an efficient and practical optimization solution for it. We have fully implemented our algorithm and report the results of experiments with both an artificial task and the standard visual adaptation benchmark for object classification, the *Office* dataset. We find that our algorithm outperforms competing approaches by producing a single robust model that performs well on any target mixture distribution.

12.2 Problem set-up

We consider a multiple-source domain adaptation problem in the general stochastic scenario where there is no unique target function is assumed for any of the domains considered and where instead there is a distribution over the joint input-output space.

Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space. We will identify a *domain* with a distribution over $\mathcal{X} \times \mathcal{Y}$ and consider the scenario where the learner has access to a predictor $h_k \colon \mathcal{X} \times \mathcal{Y} \to [0, 1]$, for each domain $\mathcal{D}_k$, $k = 1, \ldots, p$. The learner's objective is to combine these predictors to come up with an accurate predictor for a target domain that may be mixture of the source domains, or close to such mixtures.

Much of our theory applies to an arbitrary loss function $L \colon [0, 1] \to \mathbb{R}_+$ that is convex and continuous. We will denote by $\mathcal{L}(\mathcal{D}, h)$ the expected loss of a predictor h with respect

to the distribution $\mathcal{D}$:

$$\begin{aligned}\mathcal{L}(\mathcal{D},h) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} L\big(h(x,y)\big)\\ &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{D}(x,y)L\big(h(x,y)\big). \end{aligned} \tag{12.1}$$

But, we will be particularly interested in the case where L coincides with the log-loss, that is $L = -\log$. Indeed, in many applications, the predictor h_k is obtained by training (often a neural network) over a large sample drawn from a distribution $\mathcal{D}_k$ using as loss function the *cross-entropy*. The cross-entropy of $\mathcal{D}$ and a predictor h is the expected log-loss of h:

$$\begin{aligned}\mathcal{L}(\mathcal{D},h) &= -\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[\log h(x,y)\big]\\ &= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{D}(x,y)\log h(x,y). \end{aligned} \tag{12.2}$$

Observe that $\mathcal{L}(\mathcal{D},h) \geq 0$ for any $\mathcal{D}$ and h, and that $h \mapsto \mathcal{L}(\mathcal{D},h)$ is convex by convexity of $-\log$.

We will assume that each h_k is a relatively accurate predictor for the distribution $\mathcal{D}_k$ and that there exists $\epsilon > 0$ such that $\mathcal{L}(\mathcal{D}_k, h_k) \leq \epsilon$ for all $k \in [1,p]$. We will also assume that the average loss of the source predictors under the uniform distribution $\mathcal{U}$ is bounded, that is, there exists $M \geq 0$ such that

$$\frac{1}{p}\sum_{k=1}^{p} \mathcal{L}(\mathcal{U}, h_k) \leq M.$$

This is a mild assumption that is satisfied in particular when there exists $\mu > 0$ such that $h_k(x,y) \geq \mu$ for all $k \in [1,p]$ and $(x,y) \in \mathcal{X}\times\mathcal{Y}$.

12.3 Theoretical analysis

In this section, we present a theoretical analysis of the general multiple-source adaptation setting (stochastic scenario) where there is a distribution over $\mathcal{X}\times\mathcal{Y}$, as opposed to the special case where a target function mapping from $\mathcal{X}$ to $\mathcal{Y}$ is assumed (deterministic scenario) [127]. This extension is needed for the analysis of the common learning set-up where the cross-entropy loss is used and it is also more realistic for other problems.

We show that in that general setting, there exists a single weighted combination rule h_z^η that admits a small loss with respect to *any* target mixture distribution $\mathcal{D}_\lambda$, that is any λ (Section 12.3). We further give guarantees for the loss of that hypothesis with respect to an arbitrary distribution $\mathcal{D}_T$ (Section 12.3). Next, we extend our guarantees to the case where the source distributions $\mathcal{D}_k$ are not directly accessible and where the weighted combination rule is derived using estimates of the distributions $\mathcal{D}_k$ (Section 12.3).

Mixture target distribution

Here we consider the case of a target distribution that is a mixture distribution $\mathcal{D}_\lambda = \sum_{k=1}^p \lambda_k \mathcal{D}_k$, for some $\lambda = (\lambda_1, \ldots, \lambda_p)$ in the simplex $\Delta = \{\lambda \in \mathbb{R}^p \colon \forall k \in [p], \lambda_k \geq 0, \sum_{k=1}^p \lambda_k = 1\}$. The mixture weight $\lambda \in \Delta$ defining $\mathcal{D}$ is not known to us. Thus, our objective will be to find a hypothesis with small loss with respect to *any* mixture distribution $\mathcal{D}_\lambda$, using a combination of trained domain-specific hypotheses h_ks.

Following [127], we define the distribution-weighted combination of the models h_k, $k \in [1, p]$ as follows. For any $z \in \Delta$, $\eta > 0$, and $(x, y) \in \mathcal{X} \times \mathcal{Y}$,

$$h_z^\eta(x, y) = \sum_{k=1}^p \frac{z_k \mathcal{D}_k(x, y) + \eta \frac{\mathcal{U}(x,y)}{p}}{\sum_{j=1}^p z_j \mathcal{D}_j(x, y) + \eta \mathcal{U}(x, y)} h_k(x, y), \tag{12.3}$$

where $\mathcal{U}$ is the uniform distribution over $\mathcal{X} \times \mathcal{Y}$.

Observe that for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $h_z^\eta(x, y)$ is continuous in z since the denominator in (12.3) is positive ($\eta > 0$). By the continuity of L, this implies that, for any distribution $\mathcal{D}$, $\mathcal{L}(\mathcal{D}, h_z^\eta)$ is continuous in z.

Our proof makes use of the following Fixed Point Theorem of Brouwer.

Theorem 3. *For any compact and convex non-empty set $C \subset \mathbb{R}^p$ and any continuous function $f \colon C \to C$, there is a point $x \in C$ such that $f(x) = x$.*

Lemma 4. *For any $\eta, \eta' > 0$, there exists $z \in \Delta$, with $z_k \neq 0$ for all $k \in [1, p]$, such that the following holds for the distribution weighted combining rule h_z^η:*

$$\forall k \in [p], \quad \mathcal{L}(\mathcal{D}_k, h_z^\eta) \leq \sum_{j=1}^p z_j \mathcal{L}(\mathcal{D}_j, h_z^\eta) + \eta'. \tag{12.4}$$

Proof. Consider the mapping $\Phi \colon \Delta \to \Delta$ defined for all $z \in \Delta$ by

$$[\Phi(z)]_k = \frac{z_k \, \mathcal{L}(\mathcal{D}_k, h_z^\eta) + \frac{\eta'}{p}}{\sum_{j=1}^p z_j \mathcal{L}(\mathcal{D}_j, h_z^\eta) + \eta'}.$$

Φ is continuous since $\mathcal{L}(\mathcal{D}_k, h_z^\eta)$ is a continuous function of z and since the denominator is positive ($\eta' > 0$). Thus, by Brouwer's Fixed Point Theorem, there exists $z \in \Delta$ such that $\Phi(z) = z$. For that z, we can write

$$z_k = \frac{z_k \, \mathcal{L}(\mathcal{D}_k, h_z^\eta) + \frac{\eta'}{p}}{\sum_{j=1}^p z_j \mathcal{L}(\mathcal{D}_j, h_z^\eta) + \eta'},$$

for all $k \in [1, p]$. Since η' is positive, we must have $z_k \neq 0$ for all k. Dividing both sides by z_k gives $\mathcal{L}(\mathcal{D}_k, h_z^\eta) = \sum_{j=1}^p z_j \mathcal{L}(\mathcal{D}_j, h_z^\eta) + \eta' - \frac{\eta'}{p z_k} \leq \sum_{j=1}^p z_j \mathcal{L}(\mathcal{D}_j, h_z^\eta) + \eta'$, which completes the proof. □

Theorem 5. *For any $\delta > 0$, there exists $\eta > 0$ and $z \in \Delta$, such that $\mathcal{L}(\mathcal{D}_\lambda, h_z^\eta) \leq \epsilon + \delta$ for any mixture parameter $\lambda \in \Delta$.*

Proof. We first upper bound, for an arbitrary $z \in \Delta$, the expected loss of h_z with respect to the mixture distribution $\mathcal{D}_z$ defined using the same z, that is $\mathcal{L}(\mathcal{D}_z, h_z^\eta) = \sum_{k=1}^p z_k \mathcal{L}(\mathcal{D}_k, h_z^\eta)$. By definition of h_z^η and $\mathcal{D}_z$, we can write

$$\begin{aligned}
&\mathcal{L}(\mathcal{D}_z, h_z^\eta) \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{D}_z(x,y) L(h_z^\eta(x,y)) \\
&= \sum_{(x,y)} \mathcal{D}_z(x,y) L\left(\sum_{k=1}^p \frac{z_k \mathcal{D}_k(x,y) + \eta \frac{\mathcal{U}(x,y)}{p}}{\mathcal{D}_z(x,y) + \eta \mathcal{U}(x,y)} h_k(x,y)\right).
\end{aligned}$$

By convexity of $-\log$, this implies that

$$\begin{aligned}
&\mathcal{L}(\mathcal{D}_z, h_z^\eta) \\
&\leq \sum_{(x,y)} \mathcal{D}_z(x,y) \sum_{k=1}^p \frac{z_k \mathcal{D}_k(x,y) + \eta \frac{\mathcal{U}(x,y)}{p}}{\mathcal{D}_z(x,y) + \eta \mathcal{U}(x,y)} L\big(h_k(x,y)\big).
\end{aligned}$$

Next, since $\frac{\mathcal{D}_z(x,y)}{\mathcal{D}_z(x,y)+\eta\mathcal{U}(x,y)} \leq 1$, the following holds:

$$\begin{aligned}
&\mathcal{L}(\mathcal{D}_z, h_z^\eta) \\
&\leq \sum_{(x,y)} \Big(\sum_{k=1}^p \big(z_k \mathcal{D}_k(x,y) + \tfrac{\eta \mathcal{U}(x,y)}{p}\big) L(h_k(x,y)) \Big) \\
&= \sum_{k=1}^p z_k \mathcal{L}(\mathcal{D}_k, h_k) + \frac{\eta}{p} \sum_{k=1}^p \mathcal{L}(\mathcal{U}, h_k) \\
&\leq \sum_{k=1}^p z_k \epsilon + \eta M = \epsilon + \eta M.
\end{aligned}$$

Now choose $z \in \Delta$ as in the statement of Lemma 4. Then, the following holds for any mixture distribution $\mathcal{D}_\lambda$:

$$\begin{aligned}
\mathcal{L}(\mathcal{D}_\lambda, h_z^\eta) &= \sum_{k=1}^p \lambda_k \mathcal{L}(\mathcal{D}_k, h_z^\eta) \\
&\leq \sum_{k=1}^p \lambda_k (\mathcal{L}(\mathcal{D}_z, h_z^\eta) + \eta') \\
&= \mathcal{L}(\mathcal{D}_z, h_z^\eta) + \eta' \leq \epsilon + \eta M + \eta'.
\end{aligned}$$

Setting $\eta = \frac{\delta}{2M}$ and $\eta' = \frac{\delta}{2}$ concludes the proof. □

The theorem shows the existence of a mixture weight $z \in \Delta$ and $\eta > 0$ with a remarkable property: for any $\delta > 0$, regardless of which mixture weight $\lambda \in \Delta$ defining the target distribution, the loss of h^η_z is at most $\epsilon + \delta$, that is arbitrarily close to ϵ. h^η_z is therefore a *robust* hypothesis with favorable property for any mixture target distribution.

More precisely, by the proof of the theorem, for any $z \in \Delta$ verifying (12.4), h^η_z admits this property. We exploit this in the next section to devise an algorithm for finding such a $z \in \Delta$.

Arbitrary target distribution

Here, we extend the results of the previous section to the case of an arbitrary target distribution $\mathcal{D}_T$ that may not be a mixture of the source distributions by extending the results of [128].

We will assume that the loss of the source hypotheses h_k is bounded, that is $L(h_k(x, y)) \leq M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. By convexity, this immediately implies that for any distribution combination hypothesis h^η_z,

$$\begin{aligned} L(h^\eta_z(x, y)) &\leq \sum_{k=1}^{p} \frac{z_k \mathcal{D}_k(x, y) + \eta \frac{\mathcal{U}(x,y)}{p}}{\mathcal{D}_z(x, y) + \eta \mathcal{U}(x, y)} L\big(h_k(x, y)\big) \\ &\leq \sum_{k=1}^{p} \frac{z_k \mathcal{D}_k(x, y) + \eta \frac{\mathcal{U}(x,y)}{p}}{\mathcal{D}_z(x, y) + \eta \mathcal{U}(x, y)} M = M. \end{aligned}$$

Our extension to an arbitrary target distribution $\mathcal{D}_T$ is based on the divergence of $\mathcal{D}_T$ from the family of all mixtures of the source distributions $\mathcal{D}_k$, $k \in [p]$. Different divergence measures could be used in this context. The one that naturally comes up in our analysis is the *Rényi Divergence* [147]. The Rényi Divergence is parameterized by α and denote by D_α. The α-Rényi Divergence of two distributions $\mathcal{D}$ and $\mathcal{D}'$ is defined by

$$\mathsf{D}_\alpha(\mathcal{D} \parallel \mathcal{D}') = \frac{1}{\alpha - 1} \log \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{D}(x, y) \left[\frac{\mathcal{D}(x, y)}{\mathcal{D}'(x, y)} \right]^{\alpha - 1}. \tag{12.5}$$

It can be shown that the Rényi Divergence is always non-negative and that for any $\alpha > 0$, $\mathsf{D}_\alpha(\mathcal{D} \parallel \mathcal{D}') = 0$ iff $\mathcal{D} = \mathcal{D}'$, (see [5]). The Rényi divergence coincides with the following known measure for some specific values of α:

- $\alpha = 1$: $\mathsf{D}_1(\mathcal{D} \parallel \mathcal{D}')$ coincides with the standard relative entropy or KL-divergence.
- $\alpha = 2$: $\mathsf{D}_2(\mathcal{D} \parallel \mathcal{D}') = \log \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{\mathcal{D}(x,y)}{\mathcal{D}'(x,y)} \right]$ is the logarithm of the expected probabilities ratio.
- $\alpha = +\infty$: $\mathsf{D}_\infty(\mathcal{D} \parallel \mathcal{D}') = \log \sup_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \left[\frac{\mathcal{D}(x,y)}{\mathcal{D}'(x,y)} \right]$, which bounds the maximum ratio between two probability distributions.

We will denote by $\mathsf{d}_\alpha(\mathcal{D} \parallel \mathcal{D}')$ the exponential:

$$\mathsf{d}(\mathcal{D} \parallel \mathcal{D}') = e^{\mathsf{D}_\alpha(\mathcal{D}\|\mathcal{D}')} = \left[\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \frac{\mathcal{D}^\alpha(x,y)}{\mathcal{D}'^{\alpha-1}(x,y)}\right]^{\frac{1}{\alpha-1}}. \tag{12.6}$$

Given a class of distributions $\mathcal{D}$, we denote by $\mathsf{D}_\alpha(\mathcal{D} \parallel \mathcal{D})$ the infimum $\inf_{\mathcal{D}'\in\mathcal{D}} \mathsf{D}_\alpha(\mathcal{D} \parallel \mathcal{D}')$. We will concentrate on the case where $\mathcal{D}$ is the class of all mixture distributions over a set of k source distributions, i.e., $\mathcal{D} = \{\mathcal{D}_\lambda \colon \mathcal{D}_\lambda = \sum_{k=1}^p \lambda_k \mathcal{D}_k, \lambda \in \Delta\}$.

Theorem 6. *Let $\mathcal{D}_T$ be an arbitrary target distribution. For any $\delta > 0$, there exists $\eta > 0$ and $z \in \Delta$, such that the following inequality holds for any $\alpha > 1$:*

$$\mathcal{L}(\mathcal{D}_T, h_z^\eta) \leq \Big[(\epsilon + \delta)\, \mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})\Big]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}}.$$

Proof. For any hypothesis $h \colon \mathcal{X} \times \mathcal{Y} \to [0,1]$ and any distribution $\mathcal{D}$, by Hölder's inequality, the following holds:

$$\begin{aligned}
\mathcal{L}(\mathcal{D}_T, h) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{D}_T(x,y) L(h(x,y)) \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \left[\frac{\mathcal{D}_T(x,y)}{\mathcal{D}(x,y)^{\frac{\alpha-1}{\alpha}}}\right] \Big[\mathcal{D}(x,y)^{\frac{\alpha-1}{\alpha}} L(h(x,y))\Big] \\
&\leq \left[\sum_{(x,y)} \frac{\mathcal{D}_T(x,y)^\alpha}{\mathcal{D}(x,y)^{\alpha-1}}\right]^{\frac{1}{\alpha}} \left[\sum_{(x,y)} \mathcal{D}(x,y) L(h(x,y))^{\frac{\alpha}{\alpha-1}}\right]^{\frac{\alpha-1}{\alpha}}.
\end{aligned}$$

Thus, by definition of d_α, for any h such that $L(h(x,y)) \leq M$ for all (x,y), we can write

$$\begin{aligned}
&\mathcal{L}(\mathcal{D}_T, h) \\
&\leq \mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})^{\frac{\alpha-1}{\alpha}} \left[\sum_{(x,y)} \mathcal{D}(x,y) L(h(x,y))^{\frac{\alpha}{\alpha-1}}\right]^{\frac{\alpha-1}{\alpha}} \\
&= \mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})^{\frac{\alpha-1}{\alpha}} \left[\sum_{(x,y)} \mathcal{D}(x,y) L(h(x,y)) L(h(x,y))^{\frac{1}{\alpha-1}}\right]^{\frac{\alpha-1}{\alpha}} \\
&\leq \mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})^{\frac{\alpha-1}{\alpha}} \left[\sum_{(x,y)} \mathcal{D}(x,y) L(h(x,y)) M^{\frac{1}{\alpha-1}}\right]^{\frac{\alpha-1}{\alpha}} \\
&\leq \Big[\mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})\, \mathcal{L}(\mathcal{D}, h)\Big]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}}.
\end{aligned}$$

Now, by Theorem 5, there exists $z \in \Delta$ and $\eta > 0$ such that $\mathcal{L}(\mathcal{D}, h_z^\eta) \leq \epsilon + \delta$ for any mixture distribution $\mathcal{D} \in \mathcal{D}$. Thus, in view of the previous inequality, we can write, for any $\mathcal{D} \in \mathcal{D}$,

$$\mathcal{L}(\mathcal{D}_T, h_z^\eta) \leq \left[(\epsilon + \delta)\, \mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D})\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}}.$$

Taking the infimum of the right-hand side over all $\mathcal{D} \in \mathcal{D}$ completes the proof. □

Note that in the particular case where $\mathcal{D}_T$ is in $\mathcal{D}$, we have $\mathsf{d}_\alpha(\mathcal{D}_T \parallel \mathcal{D}) = 1$. For $\alpha \to +\infty$ we then retrieve the result of Theorem 5.

Arbitrary target distribution and estimate distributions

We now further extend our analysis to the case where the distributions $\mathcal{D}_k$ are not directly available to the learner and where instead estimates $\widehat{\mathcal{D}}_k$ have been derived from data.

For $k \in [p]$, let $\widehat{\mathcal{D}}_k$ be an estimate of $\mathcal{D}_k$ and define $\widehat{\epsilon}$ by

$$\widehat{\epsilon} = \max_{k \in [p]} \left[\epsilon\, \mathsf{d}_\alpha(\widehat{\mathcal{D}}_k \parallel \mathcal{D}_k)\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}}. \tag{12.7}$$

Note that when for all $k \in [p]$, $\widehat{\mathcal{D}}_k$ is a good estimate of $\mathcal{D}_k$, then $\mathsf{d}_\alpha(\widehat{\mathcal{D}}_k \parallel \mathcal{D}_k)$ is close to one and, for $\alpha \to +\infty$, $\widehat{\epsilon}$ is very close to ϵ. We will denote by $\widehat{\mathcal{D}}$ the family of mixtures of the estimates $\mathcal{D}_k$:

$$\widehat{\mathcal{D}} = \left\{ \sum_{k=1}^p \lambda_k \widehat{\mathcal{D}}_k \colon k \in \Delta \right\}. \tag{12.8}$$

We will denote by $\widehat{h}_z^\eta$ distribution weighted combination hypotheses based on the estimate distributions $\widehat{\mathcal{D}}_k$:

$$\widehat{h}_z^\eta(x, y) = \sum_{k=1}^p \frac{z_k \widehat{\mathcal{D}}_k(x, y) + \eta\, \frac{\mathcal{U}(x,y)}{p}}{\sum_{j=1}^p z_j \widehat{\mathcal{D}}_j(x, y) + \eta\, \mathcal{U}(x, y)} h_k(x, y), \tag{12.9}$$

Theorem 7. *Let $\mathcal{D}_T$ be an arbitrary target distribution. Then, for any $\delta > 0$, there exists $\eta > 0$ and $z \in \Delta$, such that the following inequality holds for any $\alpha > 1$:*

$$\mathcal{L}(\mathcal{D}_T, \widehat{h}_z^\eta) \leq \left[(\widehat{\epsilon} + \delta)\, \mathsf{d}_\alpha(\mathcal{D}_T \parallel \widehat{\mathcal{D}})\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}}.$$

Proof. By the first part of the proof of Theorem 6, for any $k \in [p]$ and $\alpha > 1$, the following inequality holds:

$$\begin{aligned}
\mathcal{L}(\widehat{\mathcal{D}}_k, h_k) &\leq \left[\mathsf{d}_\alpha(\mathcal{D}_k \parallel \widehat{\mathcal{D}}_k)\, \mathcal{L}(\mathcal{D}_k, h_k)\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}} \\
&\leq \left[\epsilon\, \mathsf{d}_\alpha(\mathcal{D}_k \parallel \widehat{\mathcal{D}}_k)\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}} \leq \widehat{\epsilon}.
\end{aligned}$$

We can now apply the result of Theorem 6 (with $\widehat{\epsilon}$ instead of ϵ and $\widehat{\mathcal{D}}_k$ instead of $\mathcal{D}_k$). In view that, there exists $\eta > 0$ and $z \in \Delta$ such that

$$\mathcal{L}(\mathcal{D}_T, h_z^\eta) \leq \left[(\widehat{\epsilon} + \delta)\, \mathsf{d}_\alpha(\mathcal{D}_T \parallel \widehat{\mathcal{D}})\right]^{\frac{\alpha-1}{\alpha}} M^{\frac{1}{\alpha-1}},$$

for any distribution $\widehat{\mathcal{D}}$ in the family $\widehat{\mathcal{D}}$. Taking the infimum over all $\widehat{\mathcal{D}}$ in $\widehat{\mathcal{D}}$ completes the proof. □

This is the most general form of our theoretical results. It shows that there exists a predictor $\widehat{h}_z^\eta$ based on the estimate distributions $\widehat{\mathcal{D}}_k$ that is $\widehat{\epsilon}$-accurate with respect to any target distribution $\mathcal{D}_T$ whose Rényi divergence with respect to the family $\widehat{\mathcal{D}}$ is not too large ($\mathsf{d}_\alpha(\mathcal{D}_T \parallel \widehat{\mathcal{D}})$ close to 1). Furthermore, $\widehat{\epsilon}$ is close to ϵ, provided that $\mathcal{D}_k$s are good estimates of $\mathcal{D}_k$s ($\mathsf{d}_\alpha(\widehat{\mathcal{D}}_k \parallel \mathcal{D}_k)$ close to 1).

Chapter 13

Continuous Adaptation

13.1 Introduction

It has become increasingly clear that there is a significant bias between available labeled visual training data and the data encountered in the real world [175, 99]. Unfortunately, supervised classifiers trained on one distribution often fail when faced with a different distribution at test time. Domain adaptation techniques offer a way to transfer information learned from source (training) data to the eventual target (test) domain, so as to diminish the performance degradation and "learn from the past." Supervised adaptation methods assume a few labeled target examples are available [104, 44, 83]. However, obtaining these is often expensive or impossible, so unsupervised adaptation is of particular importance [64, 63, 53].

In this paper, we address the problem of unsupervised adaptation to a *continuously evolving* target distribution. Specifically, we assume that

1. ample labeled data is available in the source domain,
2. the target domain examples are unlabeled and arrive sequentially,
3. the target distribution evolves over time.

One scenario where this problem occurs is object or scene classification in video streams. For example, classifying scene types in a video feed from a traffic camera is challenging. The appearance of the same scene type (class) in the target domain is constantly changing due to sunlight/shadows, time of day, sensor change to IR at nighttime, and unexpected weather patterns. Another example is classifying objects or scenes as their appearance evolves over time. These two problems are illustrated in Figure 13.1. Also, while we focus on visual tasks in this paper, the problem also occurs in spam filtering, where spammers constantly change their tactics to deceive email users, and sentiment analysis in social media. Current unsupervised domain adaptation methods cannot naturally handle such problems. They assume that the target distribution is stationary, and that a large amount of unlabeled data is available in batch for modeling this fixed target distribution.

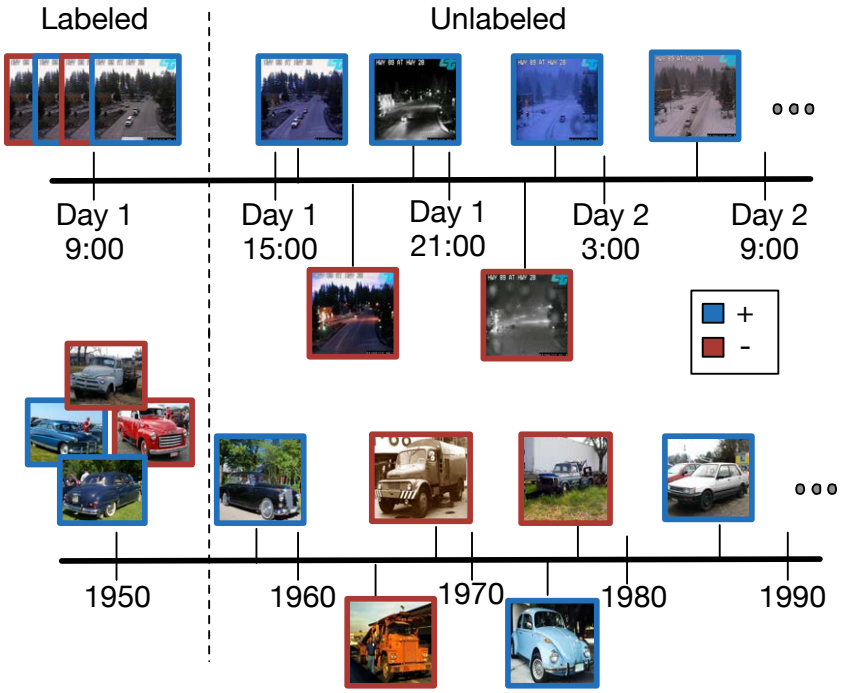


Figure 13.1: Problem setup: We want to classify test data drawn from an evolving distribution (target domain), using labeled data from a distribution collected in the past (source domain). We show two example scenarios. ABOVE: classifying traffic scenes streaming from a traffic camera as busy (blue border) or empty (red border). BELOW: classifying sedans (blue border) vs trucks (red border) across many decades as the design and shapes of the two evolve.

We stress that traditional online learning methods are not suitable for our problem. Online learning methods for classification use sequentially arriving data, but require that data to be labeled. In contrast, online distribution learning can be used for estimating an evolving domain [114, 150] , but provides no means for adapting a classifier between domains which makes it insufficient for our task. We found that learning an evolving distribution without adaptation had worse performance than classifying in the original feature space. Finally, online adaptation methods do learn from streaming observations without labels [42, 61], but expect to learn a single, stationary target distribution.

We propose a novel adaptation method which models continuously changing domain distributions by forming incremental, sample-dependent adaptive kernels. Our approach is inspired by recent methods that learn a transformation in feature space to minimize domain-induced dissimilarity [151, 104, 44, 83]. In unsupervised adaptation, this can be accomplished by projecting all source and target data points to their respective lower di-

mensional subspaces, and then minimizing the distance between the subspaces to compute a domain-invariant kernel [64, 63, 53].

However, a major limitation of these methods is that all target points are assumed to belong to a single target domain, or split into several domains with known boundaries. To apply them in our scenario, we must discretize the evolving target domain into a set of fixed domains. For the traffic camera example in Figure 13.1, this would treat all of the changes within a certain time window as a single target domain. This is problematic, as it may apply the same adaptation to, say, sunny conditions, snow storm, and night time images. A second major limitation with these methods is that data is expected in batch.

We argue that it is more natural to model the domain shift in a continuously adaptive fashion. Our technique works by learning the optimal lower dimension subspace for a specific test sample, rather than embedding it in a single monolithic subspace encompassing all of the unlabeled target data. A key advantage of our method is that there is no need to segment the test samples into a discrete set of domains, either manually or automatically, and thus no need to model the number or size of such domains. Another important advantage of our approach is the ability to more precisely adapt to each test example in an online fashion. This is helpful in situations when test samples are not available in batch but arrive sequentially. While we present an unsupervised approach, the ideas can be applied to supervised scenarios as well.

13.2 Related Work

Domain adaptation has been extensively studied in speech recognition, natural language processing and machine learning. More recently, domain adaptation techniques have been applied to visual datasets. Several supervised parameter-based adaptation methods have been proposed to learn a target classifier with a small amount of labeled training data, by regularizing the learning of a new parameter against an already learned source classifier [188, 7]. Other supervised methods learn feature transformations between source and target distributions, so classifiers may be trained directly in the source and applied to transformed target points, or trained on transformed source and transformed target data jointly [151, 104]. Some methods seek to benefit from both the discriminative power of parameter-based approaches and the flexibility of the feature-transformation approaches through unified optimization frameworks [44, 83].

A recent class of unsupervised domain adaptation techniques attempts to align the unlabeled target data with the source using manifold learning. Domains are represented as subspaces embedded in a Grassman manifold, and adaptation is carried out through geodesic flow computations on this manifold [64, 63, 53]. However, none of these methods has considered our setting of non-discrete, continuously evolving domains. They also require all unlabeled target data to be available in batch and are not designed for online adaptation. [82] argued that datasets are composed of multiple hidden domains, which they estimate via

constrained clustering, however, the number of domains is discrete and no online solution is proposed.

Supervised online learning allows a classifier to be trained with sequentially arriving data. At each round the learner receives a data point, and predicts its label. The correct answer is then revealed and the learner suffers a loss [153]. Such methods can be used to "adapt" to the incoming data stream by controlling the learning rate. In our setting, however, labels exist only in the source domain, and supervised online learning cannot be carried out.

In vision, a classic online adaptive approach is background subtraction (see [12] for a review), where the distribution of pixels belonging to the background is continuously updated. However, in classification, we are interested in categorizing the entire scene (or object), not distinguishing between foreground and background (although we can do that as a preprocessing step). In detection, a method for online adaptation was proposed that bootstraps offline classifiers to obtain new labels and uses them to continually update car detectors in a traffic scene [98, 89]. However detection fails on our traffic camera task due to the extremely low resolution of individual objects.

The natural language processing and speech recognition communities have developed algorithms to tackle the task of online adaptation. In speech recognition, the recognition of a new speaker's speech can and should be adapted and improved over time. Online incremental unsupervised fMLLR [61] dynamically collects acoustic statistics from the speaker and updates the acoustic models. [42] combines parameters of multiple classifiers to do online adaptation of spam classifiers for individual users, as well as sentiment prediction for books, movies and appliances. However, the domain change due to a new speaker or a new email user is discrete. While examples may arrive sequentially, they all arise from the same distribution (the same speaker, or the same user). On the other hand, our approach "tracks" the evolving distribution, and in that way it is somewhat akin to distribution tracking methods common in the signal processing literature, e.g., Kalman filtering [96], [117].

13.3 Continuous Adaptation Approach

Background: Unsupervised Adaptation Using the Data Manifold

We build on a class of methods recently proposed for unsupervised adaptation [64, 63, 53], which are based on modeling the data manifold. Their key insight is that visual data have an inherent low dimensional structure, and can thus be embedded in lower dimensional subspaces. Furthermore, these subspaces lie on a Grassman manifold of the same dimension. By exploiting the properties of the manifold, such as smoothness, we can find a novel embedding that compensates for the differences between domains.

Suppose we have a set of labeled examples drawn from a source domain, $\mathbf{x}_1, \ldots, \mathbf{x}_{n_\mathcal{X}} \in \mathbb{R}^D$, with labels $y_1, \ldots, y_{n_\mathcal{X}}$. At test time, we receive unlabeled examples drawn from the target domain, $\mathbf{v}_1, \ldots, \mathbf{v}_{n_\mathcal{V}} \in \mathbb{R}^D$, which are distributed differently from the source examples.

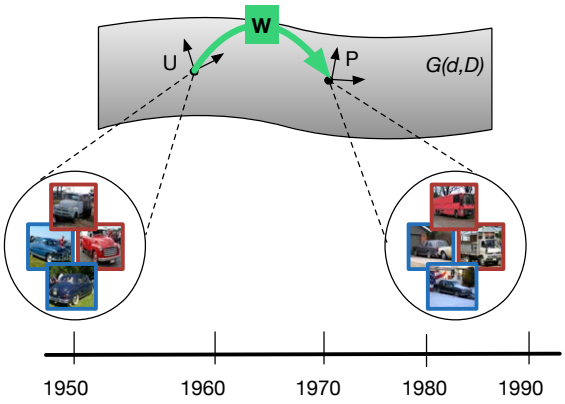


Figure 13.2: Conventional adaptation techniques separate samples into a discrete set of domains, seen here as points on a domain manifold (a single source domain S and target domain T).

For now we assume that the target examples come from a single, stationary distribution; we will relax this assumption shortly.

To account for discrepancies between the training (source) and test (target) distributions, we seek to learn a linear transformation $\boldsymbol{W}$ that maps source points in a way that makes their distribution more similar to that of the target points. Such a transformation can then be applied to compute a kernel $\mathbf{x}^T\boldsymbol{W}\mathbf{v}$, which can be used in any inner-product based classifier. An alternative is to factor the transformation into two transformations $\boldsymbol{W} = AB^T$, where A and B embed source and target points, respectively, in a new subspace.

To find $\boldsymbol{W}$, we assume the source and target domains lie on lower dimensional orthonormal subspaces, $\boldsymbol{U}, \boldsymbol{P} \in \mathbb{R}^{D\times d}$, which are points on the Grassman manifold, $\mathcal{G}(d, D)$ (See Fig. 13.2), where $d \ll D$. Several techniques exist for finding such low dimensional embeddings, including Principal Component Analysis. We then reformulate our goal as finding embeddings $\tilde{A}$ and $\tilde{B}$ that map the low dimensional subspaces in such a way as to make them better aligned. This objective can be formalized as minimizing the distance between the two projected subspaces, $\boldsymbol{U}\tilde{A}$ and $\boldsymbol{P}\tilde{B}$.

$$\min_{\tilde{A},\tilde{B}} \quad \psi(\boldsymbol{U}\tilde{A}, \boldsymbol{P}\tilde{B}) \tag{13.1}$$

One recent approach, called the Subspace Alignment (SA) method [53], solves the unconstrained optimization problem in Equation (13.1) directly, setting the subspace distance metric to be the Frobenius norm difference: $\psi(\boldsymbol{U}\tilde{A}, \boldsymbol{P}\tilde{B}) = \|\boldsymbol{U}\tilde{A} - \boldsymbol{P}\tilde{B}\|_F^2$. Since both $\boldsymbol{U}$ and $\boldsymbol{P}$ are orthonormal matrices, the global minimizer for this subspace distance metric is reached when $\tilde{A} = \boldsymbol{U}^T\boldsymbol{P}$ and $\tilde{B} = I$. This leads to the following transformation between points in the original spaces: $\boldsymbol{W}_{\text{SA}} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{P}\boldsymbol{P}^T$.

Another recent method that seeks to find embeddings for the source and target points, so as to minimize the distance between their distributions, is the Geodesic Flow Kernel

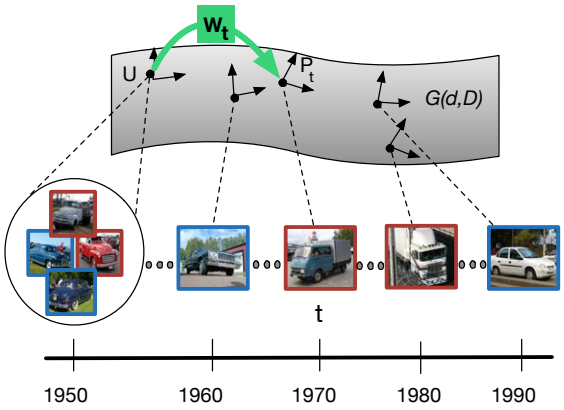


Figure 13.3: Our approach (CMA) treats each target sample as arising from a different point (ex: indexed by time) along the continuous domain manifold, resulting in more precise adaptation.

(GFK) [63]. This method learns a symmetric embedding ($A = B$) by computing the geodesic flow along the manifold, $\boldsymbol{\phi}(\cdot)$. The flow is constructed in such a way that it starts at the source subspace at time 0, $\boldsymbol{U} = \boldsymbol{\phi}(0)$, then reaches the target subspace in unit time: $\boldsymbol{P} = \boldsymbol{\phi}(1)$. The intuition is to project all source and target points into all intermediate subspaces along the flow between the source and target subspace. The final transformation is then computed by integrating over the infinite set of all such intermediate subspaces between the source and target $\boldsymbol{W}_{\text{GFK}} = \int_0^1 \boldsymbol{\phi}(\ell)\boldsymbol{\phi}(\ell)^T d\ell$, which has a closed form solution presented in [146, 63].

Adapting to Continuously Evolving Domains

We seek to adapt to and classify streaming target data that is drawn from a continuously evolving distribution. The drawback of the above methods is that they require discrete known domains, where the data from each domain is available in batch (see Figure 13.2). To adapt to each instance the above methods would need to artificially discretize the target by using a fixed windowed history and would still fail to adapt until enough data had arrived to begin learning subspaces. This is not what the method was originally designed for, would be very computationally expensive and would require cross-validating or tuning a hyper-parameter to choose the appropriate window size. Next, we present our approach, Continuous Manifold Adaptation (CMA), which does not require knowledge of discrete domains (see Figure 13.3).

Suppose that at test time, we receive a stream of observations $\mathbf{v}_1, \ldots, \mathbf{v}_{nT} \in \mathbb{R}^D$, which arrive one at a time, and are drawn from a continuously changing domain.[1] We assume the distribution of possible points arriving at t can be represented by a lower dimensional subspace $\boldsymbol{P}_t$.

[1]Our formulation can also be extended to the case of streaming source observations.

To align the training and test data, we seek to learn a time-varying transformation, $\boldsymbol{W}_t$, between source and target points, where t indexes the order in which the examples are received. As presented in Section 13.3, this transformation can equivalently be written as learning two time-varying embeddings that map between points of the two lower dimensional subspaces, $\tilde{A}_t$ and $\tilde{B}_t$, with the mapping in the original space being defined as $\boldsymbol{W}_t = \tilde{A}^T \boldsymbol{U}^T \boldsymbol{P}_t \tilde{B}_t$. This computes a time varying kernel between the source data and the evolving target data $\mathbf{x}^T \boldsymbol{W}_t \mathbf{v}_t$ which can be used with any inner product based classifier.

Since we no longer have a fixed target distribution with all examples delivered in batch, we must simultaneously learn the lower dimensional subspace, $\boldsymbol{P}_t$, representing the distribution from which the data was drawn at each time t. We will search for a subspace that minimizes the re-projection error of the data:

$$R_{err}(\mathbf{v}_t, \boldsymbol{P}_t) \quad = \quad \|\mathbf{v}_t - \boldsymbol{P}_t(\boldsymbol{P}_t^T \mathbf{v}_t)\|_F^2 \tag{13.2}$$

In general, we may receive as few as one data point at each time step so we will regularize our subspace learning by a smoothness assumption that the target subspace does not change quickly.[2]

Therefore, at each time step, our goals can be summarized by optimizing the following problem:

$$\min_{\boldsymbol{P}_t^T \boldsymbol{P}_t = I, \tilde{A}_t, \tilde{B}_t} r(\boldsymbol{P}_{t-1}, \boldsymbol{P}_t) + R_{err}(\mathbf{v}_t, \boldsymbol{P}_t) + \psi(\boldsymbol{U}\tilde{A}_t, \boldsymbol{P}_t \tilde{B}_t) \tag{13.3}$$

where $r(\cdot)$ is a regularizer that encourages the new subspace learned at time t to be close to the previous subspace of time $t-1$.

Equation (13.3) is a non-convex problem and we choose to solve it by alternating between the three steps below:

1. Receive data $\mathbf{v}_t$
2. Given $\tilde{A}_{t-1}$ and $\tilde{B}_{t-1}$ compute $\boldsymbol{P}_t$
3. Given $\boldsymbol{P}_t$ compute $\tilde{A}_t$ and $\tilde{B}_t$

To optimize step 2, we begin by fixing $\tilde{A}_{t-1}$ and $\tilde{B}_{t-1}$ and then we examine the third term of the optimization function. Note that it would be minimized if $\boldsymbol{P}_t = \boldsymbol{P}_{t-1}$. Therefore, with a fixed $\tilde{A}_{t-1}$ $\tilde{B}_{t-1}$, the term $\psi(\boldsymbol{U}\tilde{A}_{t-1}, \boldsymbol{P}_t \tilde{B}_{t-1})$ is acting as a regularizer that penalizes when $\boldsymbol{P}_t$ deviates from $\boldsymbol{P}_{t-1}$. We therefore can equivalently solve this problem by grouping the first and third term into a single regularizer of $\boldsymbol{P}_t$ that enforces a smoothness between the subsequent learned subspaces. Finally, we can express this subproblem as follows:

$$\begin{aligned} \min_{\boldsymbol{P}_t} \quad & r(\boldsymbol{P}_{t-1}, \boldsymbol{P}_t) + R_{err}(\mathbf{v}_t, \boldsymbol{P}_t) \\ \text{s.t} \quad & \boldsymbol{P}_t^T \boldsymbol{P}_t = I \end{aligned} \tag{13.4}$$

[2]Our model can be extended to allow for discontinuities, but we leave this as future work.

We first observe that solving Equation (13.4) for the trivial regularizer $r(\cdot, \cdot) = \text{constant}$ would result in $\boldsymbol{P}_t$ which is equal to the d largest singular vectors of the data $\mathbf{v}_t$, which can be obtained via SVD. Obviously, we prefer to use a non-trivial regularizer, as we don't have enough data at time t to compute a robust SVD, and also want to make sure that the subspaces vary smoothly over time. Thus we solve this optimization problem with a variant of sequential Karhunen-Loeve [114], which adapts a subspace incrementally and trades-off changing the subspace with minimizing re-projection error of $\mathbf{v}$. For optimization details see [150].

When optimizing step 3, we note that the first two terms of the objective function are not active for this sub-problem, and we are left with the task of minimizing Equation (13.1).

This is just the task of aligning two known subspaces. We experiment with solving this optimization using either of the two methods described in Section 13.3.

13.4 Experiments and Results

We present performance on both a scene classification experiment and an object classification experiment. For both experiments we compare our Continuous Manifold Adaptation (CMA) approach using two different unsupervised adaptation techniques (Geodesic flow kernel (GFK)[63] and Subspace Alignment method (SA)[53]) for solving Step 3 in our algorithm and two different inner product based classifiers: k-nearest neighbors (KNN with $k = 1$) and support vector machines (SVM) – trained with source data only. We demonstrate performance increase using our CMA method across a variety of feature spaces for these tasks. The unsupervised adaptation methods can not be directly applied to our problem with the streaming test domain. However, for completeness we tried learning subspaces from a fixed windowed history and then used the unsupervised adaptation approaches. We ran experiments evaluating the performance of various window sizes (including using all history available, which is computationally infeasible in practice), but were unable to find a result that was competitive with our method.

Scene Classification Over Time

Dataset Our first experiment evaluates our algorithm on scene classification over time using a real-world surveillance dataset. The images were captured from a fixed traffic camera observing an intersection. Frames were updated at 3 minute intervals each with a resolution of 320x240[3]. Our dataset includes images captured over a 2 week period. This data offers a challenging domain shift problem as changes include illumination, shadows, fog, snow, light saturation from oncoming sedans, change to night time IR mode, etc.

Experiment Setup We define an intersection traffic classification task, which is to determine whether one or more cars are present in, or approaching, the intersection. We obtain

[3]Available at the California Department of Transportation website, `http://quickmap.dot.ca.gov/`

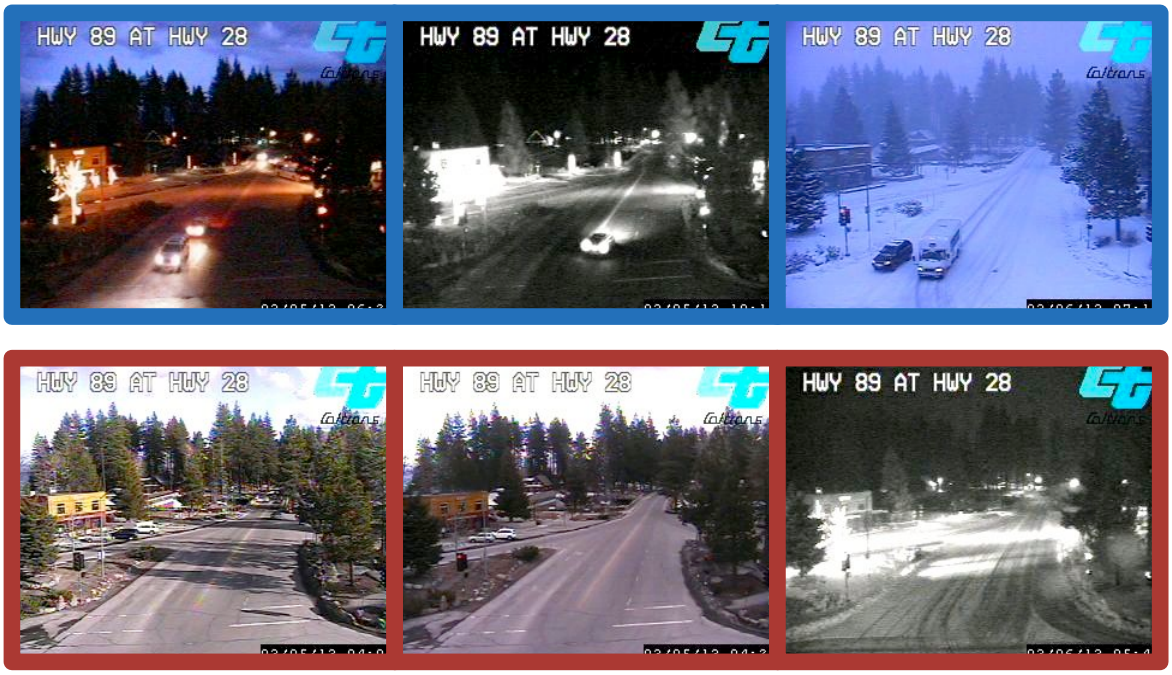


Figure 13.4: Sample human labeled images used for intersection traffic classification. Positive examples are shown in the top row (blue) and negative examples are shown in the bottom row (red).

labels for this task using human annotators (for example labels see Figure 13.4). We assume to be given 50 labeled consecutive images (2.5 hours) and then evaluate each algorithm on the immediately following 24 hours (480 images) and 5 days (2400 images). We evaluate the classifiers in the online setting, where classification must occur just after receiving a test point and may only be informed by previously received test data with no knowledge of future test data.

This task is challenging and cannot be adequately solved with approaches such as scanning-window car detection, as the images (and especially the cars within) are too low-resolution to be detected by conventional methods. A deformable parts model (DPM) detector [52] failed to detect any sedans in the first 50 images. Instead we compute features over the whole image and produce a scene label.

We consider two features that are known to perform well on scene classification tasks: GIST [157] (512 dimension) and SIFT-SPM [110] using a 200 dimension codebook and 3 pyramid layers (4200 dimension). Finally, since the images are all of a fixed scene we use a standard mixture of gaussians background subtraction algorithm [168] to extract a foreground mask and compute the same GIST and SIFT-SPM on the foreground. We found that sequential images were far too noisy to provide useful foreground masks; we present all

Adaptation Method	Classifier	GIST[157]	SIFT-SPM[110]	GIST[157] + BSub[168]	SIFT-SPM[110]+BSub[168]
-	KNN	76.30± 3.0	47.51±5.1	52.27±3.4	39.91±3.0
-	SVM	74.42± 3.0	68.69±3.6	50.98±3.6	48.91±3.0
CMA+GFK	KNN	**78.07±1.8**	49.84±5.5	52.97±2.7	39.08±2.6
CMA+GFK	SVM	**78.38± 3.1**	74.98±2.7	59.55±2.9	47.59±2.8
CMA+SA	KNN	**78.71±1.7**	54.08±6.2	51.33±4.2	38.21±2.6
CMA+SA	SVM	**78.49±3.1**	75.66±2.9	59.68±2.9	49.05±2.8

Table 13.1: Our method, CMA, improves performance independent of the feature choice for the scene classification task. Results here are shown with optimizing the unsupervised adaptation problem using either the geodesic flow kernel (GFK)[63] or the subspace alignment (SA) method [53]. Average precision (%) is recorded when training with 50 labeled images and testing on the immediately following 24 hours (480 images).

Adaptation	Classifier	GIST[157]
-	KNN	71.24±5.7
-	SVM	80.40±0.6
CMA+GFK	KNN	77.21± 3.8
CMA+GFK	SVM	**84.17±1.5**
CMA+SA	KNN	78.61±3.3
CMA+SA	SVM	**84.32±1.4**

Table 13.2: Our method, CMA, continues to provide improvement for the scene classification task even when testing over the 5 days following the labeled training data. We show here average precision (%) for the 2400 test images following the 50 available labeled training images.

results here for completeness.

Results & Analysis Table 13.1 presents the average precision (%) when testing on the 24 hours immediately following the labeled data. CMA is shown to provide improvement over no-adaptation regardless of feature choice. The strongest algorithm and feature combination for this setup was to use CMA with GIST features and either type of subspace alignment algorithm and either classifier.

We next demonstrate that the algorithm does not diverge and in fact continues to provide improvement by testing over the a 5 day period (see Table 13.2). Here we show results using the GIST feature with each type of classifier and adaptation optimization algorithm. We found that SVM generalized better over time.

To understand the performance of the adaptive method, we examine qualitative classifi-

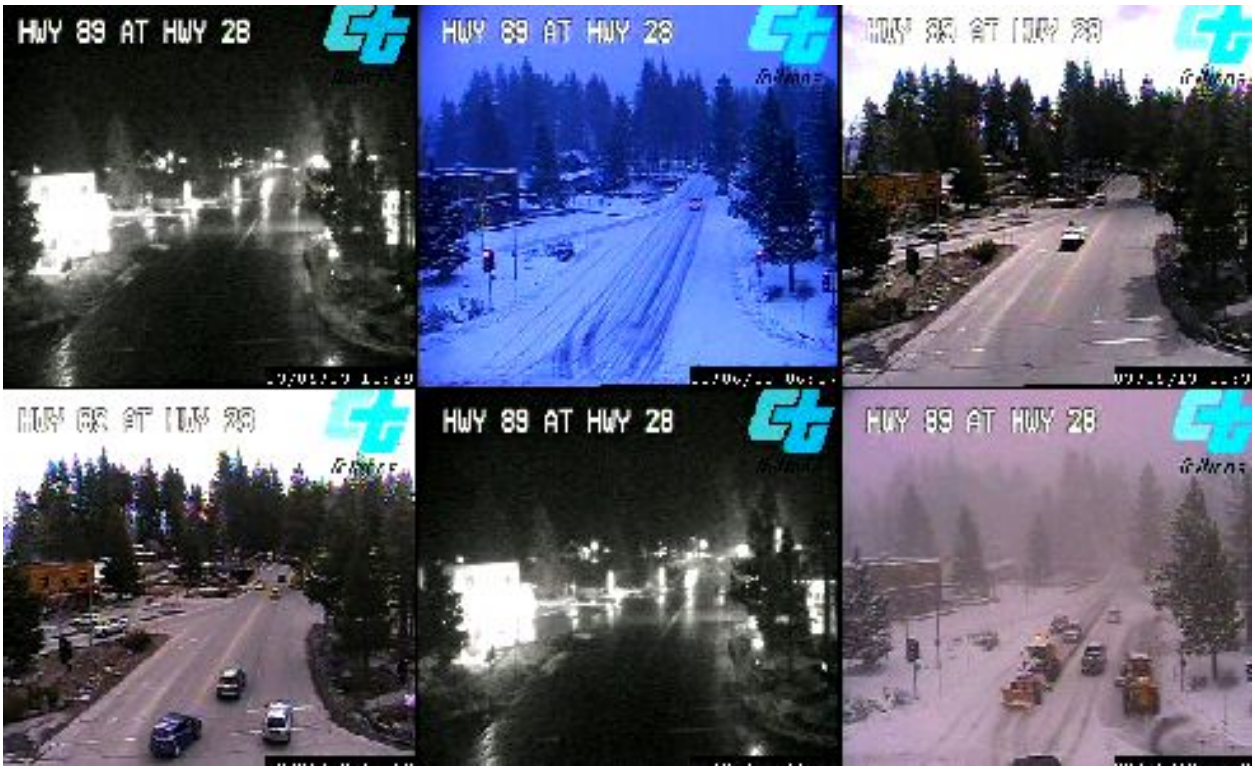


Figure 13.5: Qualitative results from the intersection traffic classification task. Training on day-time images with no snow only. Images labeled correctly by our method (CMA) and incorrectly labeled by all other methods. We show here the 6 examples for which the baseline had highest (incorrect) confidence, indicating that these examples were particularly challenging for the baseline and then fixed with our method. We improve in the cases of nighttime, snow, and fog, not seen during training.

cation examples. Figure 13.5 shows images that were misclassified by all algorithms except our CMA approach. The sedans parked in the parking lot on the left side of the image as well as the protrusion from the snow mound between the road and turn-out were likely confusions for the non-adaptive baselines. Figure 13.6 shows images incorrectly classified by all algorithms. Here are negative examples that may have sedans present, but too far away to be considered traffic at the intersection by our task definition.

For reference, if one had access to all of the test data in **batch** one could directly apply an adaptation methods or even pre-cluster the test data and learn multiple transformations. The performance for batch test data using GIST features with SA and SVM is 76.44 AP for the single cluster case and 77.57 AP for the multiple cluster case. These are both for the 1 day test set. We see here that actually our algorithm is performing even better than using the algorithms in batch with pre-clustering of the data.

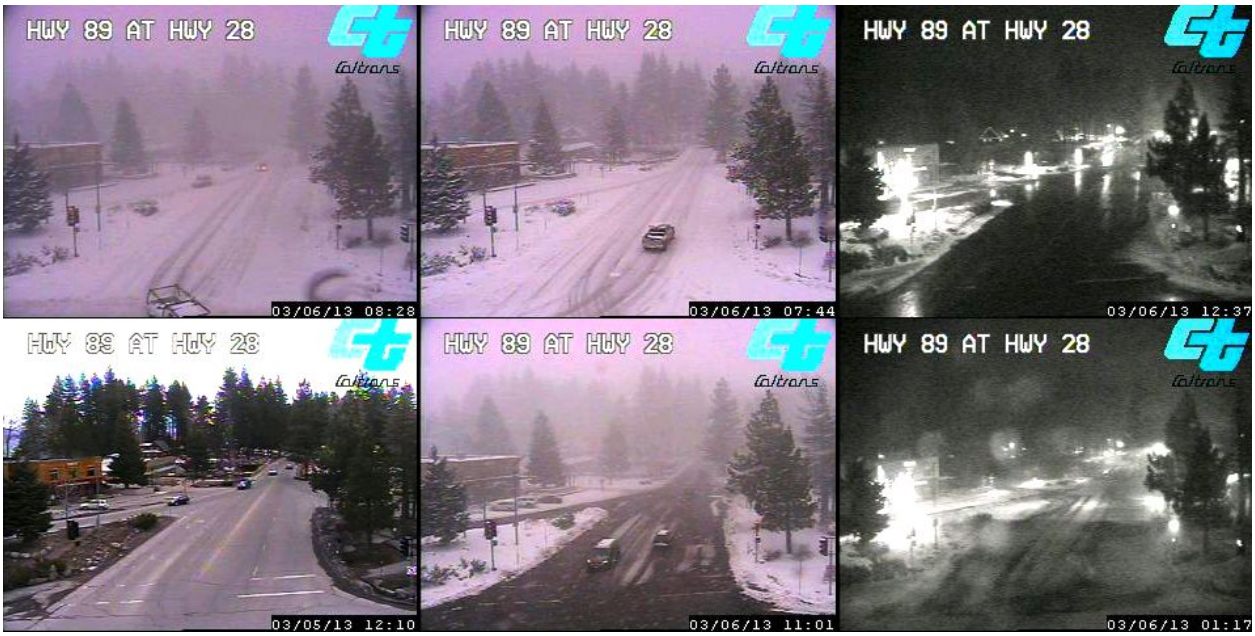


Figure 13.6: Qualitative results from the intersection traffic classification task. Example images where all methods classified incorrectly – snow, sedans too far away, and bright lights in the distance make these images difficult.

Object Classification Over Time

Dataset Next, we evaluate on the task of distinguishing sedans and trucks over time. We collected a new automobile dataset that contains images of automobiles manufactured between the years of 1950-2000. The data was acquired from a freely available online database[4] that has object centric images of automobiles, each user labeled with a manufactured year and a model label. This database was recently proposed for detecting connections in space and time [112]. The images vary in size but are usually around 400x600. We collected 30-40 images (depending on availability) from each year of the images that were tagged as either a sedan or a truck. We directly used those tag labels as our ground truth for the car and truck classes. See Figure 13.1 (bottom row) for example images.

Experiment Setup Our task is to classify each test image as either a car or a truck. We use the first 5 years of data (1950-1954) as our labeled source examples. We then consider receiving all subsequent test data sequentially in time. As in the previous experiment we use both GIST [157] (512 dimension) and SIFT-SPM [110] using a 200 dimension codebook and 3 pyramid layers (4200 dimension) representations for this data. Additionally, as this is an object classification task, we also experiment with a recently proposed feature based

[4] `http://www.cardatabase.net`

Adaptation	Classifier	SIFT-SPM [110]	GIST [157]	DeCAF [40]
-	KNN	66.31± 0.6	72.77± 0.8	84.60± 0.7
-	SVM	79.26± 0.6	76.40± 0.7	85.92± 0.4
CMA+GFK	KNN	66.32± 0.2	72.60± 0.9	82.65± 0.5
CMA+GFK	SVM	80.24± 0.7	78.32± 0.6	**89.68± 0.1**
CMA+SA	KNN	65.06± 1.1	71.44± 1.3	81.97± 0.6
CMA+SA	SVM	79.79± 0.6	78.31± 0.7	**89.71± 0.1**

Table 13.3: Our algorithm improves performance on category recognition task. We evaluate our continuous manifold adaptation approach (CMA) on the task of labeling images of automobiles as either cars or trucks. We show results using two solutions to the unsupervised adaptation problem (GFK[63] and SA[53]) and two inner product based source classifiers (KNN and SVM). We compare across three types of features and demonstrate the benefit of using our algorithm for each feature choice, including a deep learning based feature that was tuned for object classification on all of ImageNet[40].

on vectorizing a layer of a deep learning architecture trained on all of ImageNet, called DeCAF [40].[5]

Results & Analysis We present classification accuracy results on the automobile dataset in Table 13.3. All results represent an average across 10 random train/test splits. Our algorithm, CMA, provides a significant accuracy improvement over the non-adaptive baselines for the GIST and DeCAF features. The best overall results, with 90% accuracy, were achieved using the DeCAF features and our CMA approach followed by an SVM classifier.

To get a sense for which examples CMA provides improvement, we looked at the set of images that were incorrectly classified by a non-adaptive source SVM and then were correctly classified by CMA. We then displayed the 5 car and 5 truck examples for which the SVM has the highest (incorrect) confidence – indicating these were difficult examples (see Figure 13.7). In particular, they include sedans on top of trucks and trucks with ramps off the back.

There were also examples for which all methods misclassified the results (see Figure 13.8). All algorithms were consistently confused by vans and pickup trucks with covered beds, labeling them as sedans (though it's debatable which category the vans should fall into anyway). Additionally, sedans with distinctive front grates or high profiles sedans were sometimes confused with trucks. There were in general more mislabeled trucks than sedans.

[5]For our experiments we use the vectorized output of layer 6 of the network.

Figure 13.7: Our method clearly adapts to vehicle appearance as it evolves to look different from that in the labeled 50's training data. We show example images misclassified by non-adaptive SVM (DeCAF features) and correctly classified by CMA followed by the same SVM classifier. The 5 sedans and 5 trucks for which the SVM had the highest confidence (though incorrect) are displayed here.

13.5 Conclusion

We have presented a novel problem statement of performing a visual classification task under dynamic distribution shift. Our solution method dynamically learns data specific subspaces through time in order to compute an adaptive transformation at each time step. We experimentally validate that our algorithm outperforms non-adaptive baselines, independent of feature representation, and across two real world visual adaptation tasks where the target is dynamically distributed over time.

We focused on the unsupervised learning task because of its practical importance, but in the future we would like to examine the performance benefit of adding a few labeled target examples in an active learning framework. We suspect that especially in the setting where there are sudden dramatic shifts in the data, the discrepancy is perceptible by the algorithm and some supervision could focus the subspace learning and boost performance.

Figure 13.8: Example images misclassified by all methods (sedans top and trucks bottom). Vans and trucks with covered beds were consistently labeled as sedans by all algorithms. Additionally, sedans with distinctive front grates and/or high profiles were sometimes confused with trucks.

Part V

Adapting Across Tasks

Chapter 14

Introduction and Background

14.1 Problem Setup

It is well known that contemporary visual models thrive on large amounts of training data, especially those that directly include labels for the desired tasks. Many real world settings contain labels with varying specificity, e.g., "strong" bounding box detection labels, and "weak" labels indicating presence somewhere in the image. We tackle the problem of *joint detector and representation learning*, and develop models which cooperatively exploit heterogeneous sources of training data, where some classes have no "strong" annotations. Our model optimizes a latent variable multiple instance learning model over image regions while simultaneously transferring a shared representation from detection-domain models to classification-domain models. The latter provides a key source of automatic and accurate initialization for latent variable optimization, which has heretofore been unavailable in such methods.

Both classification and detection are key visual recognition challenges, though historically very different architectures have been deployed for each. Recently, the R-CNN model [59] showed how to adapt an ImageNet classifier into a detector, but required bounding box data for all categories. We ask, is there something generic in the transformation from classification to detection that can be learned on a subset of categories and then transferred to other classifiers?

One of the fundamental challenges in training object detection systems is the need to collect a large of amount of images with bounding box annotations. The introduction of detection challenge datasets, such as PASCAL VOC [49], has propelled progress by providing the research community a dataset with enough fully annotated images to train competitive models although only for 20 classes. Even though the more recent ILSVRC13 detection dataset [35] has extended the set of annotated images, it only contains data for 200 categories. The larger ImageNet dataset contains some localization information for around 3000 object categories, though these are not exhaustively labeled. As we look forward towards the goal of scaling our systems to human-level category detection, it becomes impractical to collect

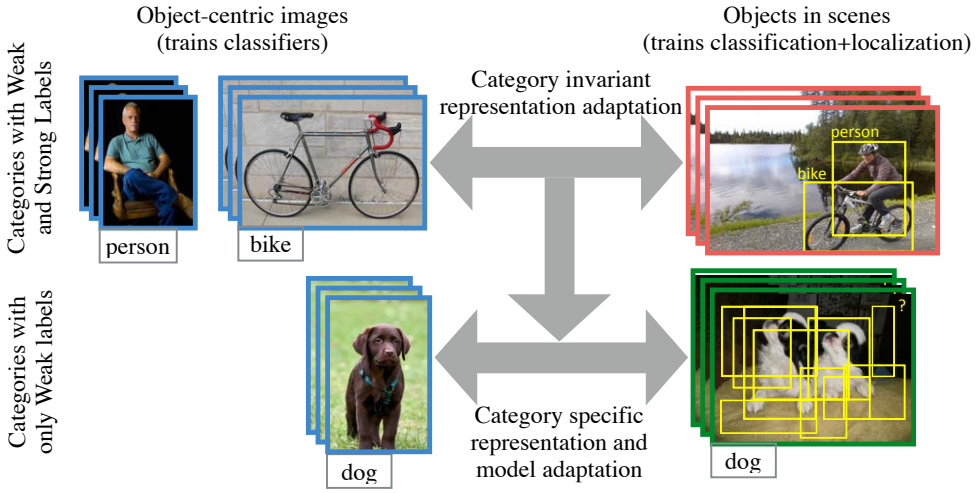


Figure 14.1: We learn detectors (models which classify and localize) for categories with only weak labels (*bottom row*). We use auxiliary categories with available paired strong and weak annotations (*top row*) to learn to adapt a visual representation from whole image classification to localized region detection. We then use the adapted representation to transform the classifiers trained for the categories with only weak labels and jointly solve an MIL problem to mine localized training data from the weakly labeled scene-centric training data (*green – bottom right*).

a large quantity of bounding box labels for tens or hundreds of thousands of categories.

In contrast, image-level annotation is comparatively easy to acquire. The prevalence of image tags allows search engines to quickly produce a set of images that have some correspondence to any particular category. ImageNet [15], for example, has made use of these search results in combination with manual outlier detection to produce a large classification dataset comprised of over 20,000 categories. While this data can be effectively used to train object classifier models, it lacks the supervised annotations needed to train state-of-the-art detectors.

Previous methods employ varying combinations of weak and strong labels of the same object category to learn a detector. Such methods seldom exploit available strong-labeled data of different, auxiliary categories, despite the fact that such data is very often available in many practical scenarios. [36] uses auxiliary data to learn generic objectness information just as an initial step, but doesn't optimize jointly for weakly labeled data.

We introduce a new model for large-scale learning of detectors that can jointly exploit weak and strong labels, perform inference over latent regions in weakly labeled training examples, and can transfer representations learned from related tasks (see Figure 14.1). In practical settings, such as learning visual detector models for all available ImageNet categories, or for learning detector versions of other defined categories such as Sentibank's

adjective-noun-phrase models [21], our model makes greater use of available data and labels than previous approaches. Our method takes advantage of such data by using the auxiliary strong labels to improve the feature representation for detection tasks, and uses the improved representation to learn a stronger detector from weak labels in a deep architecture.

We cast the task as a domain adaptation problem, considering the data used to train classifiers (images with category labels) as our source domain, and the data used to train detectors (images with bounding boxes and category labels) as our target domain. We then seek to find a general transformation from the source domain to the target domain, that can be applied to any image classifier to adapt it into a object detector (see Figure 14.1). R-CNN [59] demonstrated that adaptation, in the form of fine-tuning, is very important for transferring deep features from classification to detection and partially inspired our approach. However, the R-CNN algorithm uses classification data only to pre-train a deep network and then requires a large number of bounding boxes to train each detection category.

To learn detectors, we exploit weakly labeled data for a concept, including both object-centric images (e.g., from ImageNet classification training data), and weakly labeled scene-centric imagery (e.g., from PASCAL or ImageNet detection training data with bounding box metadata removed). We define a novel multiple instance learning (MIL) framework that includes bags defined on both types of data, and also jointly optimizes an underlying perceptual representation using strong detection labels from related categories. We demonstrate that a good perceptual representation for detection tasks can be learned from a set of paired weak and strong labeled examples and the resulting adaptation can be transferred to new categories, even those for which no strong labels were available.

We additionally show that our large-scale detection models can be directly converted into models which produce pixel-level localization for each category. Following the recent result of [123], we run our models fully-convolutionally and directly use the learned detection weights to predict per-pixel labels.

We evaluate our detection model empirically on the largest set of available ground-truth visual data labeled with bounding box annotations, the ILSVRC13 detection dataset. Our method outperforms the previous best MIL-based approaches for weakly labeled detector learning [183] on ILSVRC13 [35] by 200%. Our model is directly applicable to learning improved "detectors in the wild", including categories in ImageNet but not in the ILSVRC13 detection dataset, or categories defined ad-hoc for a particular user or task with just a few training examples to fine-tune a new classification model. Such models can be promoted to detectors with no (or few) labeled bounding boxes.

14.2 Related Work

Since its inception, the multiple instance learning (MIL) problem [38], or learning from a set of labels that specify at least one instance in a bag of instances, has been attempted in several frameworks, including Noisy-OR and boosting [2, 195]. However, most commonly, it

has been framed as a max-margin classification problem [3], with latent parameters optimized using alternating optimization [52, 191].

Recently, MIL has also been used in computer vision to train detectors using weak labels, i.e. images with category labels but without bounding box labels. The MIL paradigm estimates latent labels of examples in positive training bags, where each positive bag is known to contain at least one positive example. For example, [56] and [2] construct positive bags from all object proposal regions in a weakly labeled image that is known to contain the object and use a version of MIL to learn an object detector. Overall, MIL is tackled in two stages: first, finding a good initialization, and second, using good heuristics for optimization. A number of methods have been proposed for initialization which include using a large image region excluding boundary [139], using a candidate set which covers the training data space [163, 164], using unsupervised patch discovery [162, 160], learning generic objectness knowledge from auxiliary categories [1, 36], learning latent categories from background to suppress it [183], or using class-specific similarity [161]. Approaches to better optimize the non-convex problem involve using multi-fold learning as a measure of regularizing overfitting [28], optimizing Latent SVM for the area under the ROC curve (AUC) [17], and training with easy examples initially to avoid bad local optima [14, 107, 66].

While these approaches are promising, they often underperform on the full detection task in more challenging settings such as the PASCAL VOC dataset [49], where objects only cover small portions of images, and many candidate bounding boxes contain no objects whatsoever. The major challenges faced by solutions to the MIL problem are the limitations of fixed feature representations and poor initializations, particularly in non-object centric images. Our algorithm provides solutions to both of these issues. We also provide an evaluation on the large-scale ILSVRC13 detection dataset, which many previous methods have not been evaluated on.

Deep convolutional neural networks (CNNs) have emerged as state of the art on popular object classification benchmarks such as ILSVRC [103] and MNIST. In fact, "deep features" extracted from CNNs trained on the object classification task are also state of the art on other tasks such as subcategory classification, scene classification, domain adaptation [40], and even image matching [54]. Unlike the previously dominant features (SIFT [125], HOG [31]), deep CNN features can be learned for each specific task, but only if sufficient labeled training data is available. R-CNN [59] showed that fine-tuning deep features, pre-trained for classification, on a large amount of bounding box labeled data significantly improves detection performance.

Domain adaptation methods aim to reduce dataset bias caused by a difference in the statistical distributions between training and test domains. In this paper, we treat the transformation of classifiers into detectors as a domain adaptation task. Many approaches have been proposed for classifier adaptation, such as feature space transformations [151, 104, 63, 53], model adaptation approaches [188, 7], and joint feature and model adaptation [83, 40]. However, even the joint learning models are not able to modify the feature extraction process and so are limited to shallow adaptation techniques. Additionally, these methods only adapt between visual domains, keeping the task fixed, while we adapt both from a large

visual domain to a smaller visual domain and from a classification task to a detection task.

However, domain adaptation techniques have seen recent success through the merger with deep CNNs. [76] showed that, when training data in the target domain is severely limited or unavailable, domain adaptation techniques as applied to CNNs can be more effective than the standard practice of fine-tuning. More recent works have seen success in augmenting deep architectures with additional regularization layers that are robust to the negative effects of domain shift [58, 176, 124, 57]. However, all of these methods focus on the standard visual domain adaptation problem, where one adapts between two versions of the same task with different statistics, and do not investigate the task adaptation setting.

Several supervised domain adaptation models have been proposed for object detection. Given a detector trained on a source domain, they adjust its parameters on labeled target domain data. These include variants for linear support vector machines [189, 7, 41], as well as adaptive latent SVMs [186] and adaptive exemplar SVM [6]. A related recent method [62] proposes a fast adaptation technique based on Linear Discriminant Analysis. These methods require strongly labeled data with bounding box annotations for all object categories, both in the source and target domains, which is absent in our scenario.

Other methods have been proposed that use the underlying semantic hierarchy of ImageNet to transfer localization information to classes for strong labels are available [65, 180]. However, this necessarily limits their approaches to settings in which additional semantic information is available.

Background: MIL

We begin by briefly reviewing a standard solution to the multiple instance learning problem, Multiple Instance SVMs (MI-SVMs) [3] or Latent SVMs [52, 191]. In this setting, each weakly labeled image is considered a collection of bounding boxes which form a positive 'bag'. For a binary classification problem, the task is to maximize the bag margin which is defined by the instance with highest confidence. For each weakly labeled image $I \in \mathcal{W}$, we collect a set of bounding boxes and define the index set of those boxes as R_I. We next define a bag as $B_I = \{\boldsymbol{x}_i | i \in R_I\}$, with label Y_I, and let the i^{th} instance in the bag be $(\boldsymbol{x}_i, y_i) \in \mathcal{R}^p \times \{-1, +1\}$.

For an image with a negative image-level label, $Y_I = -1$, we label all bounding boxes in the image as negative. For an image with a positive image-level label, $Y_I = 1$, we create a constraint that at least one positive instance occurs in the image bag.

In a typical detection scenario, R_I corresponds to the set of possible bounding boxes inside the image, and maximizing over R_I is equivalent to discovering the bounding box that contains the positive object. We define a representation $\phi(\boldsymbol{x}_i) \in \mathcal{R}^d$ for each instance, which is the feature descriptor for the corresponding bounding box, and formulate the MI-SVM objective as follows:

$$\min_{\boldsymbol{w} \in \mathcal{R}^d} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \alpha \sum_I \ell\Big(Y_I, \max_{i \in R_I} \boldsymbol{w}^T \phi(\boldsymbol{x}_i)\Big) \tag{14.1}$$

where α is a hyper-parameter and $\ell(y, \hat{y})$ is the hinge loss. Interestingly, for negative bags i.e. $Y_I = -1$, the knowledge that all instances are negative allows us to unfold the max operation into a sum over each instance. Thus, Equation (14.1) reduces to a standard QP with respect to $\boldsymbol{w}$. For the case of positive bags, this formulation reduces to a standard SVM if the maximum scoring instance is known.

Based on this idea, Equation (14.1) is optimized using a classic concave-convex procedure [193], which decreases the objective value monotonically with a guarantee to converge to a local minima or saddle point. Due to this reason, weakly trained MIL detectors are sensitive to the feature representation and initial detector weights (i.e. initialization in MIL) [28, 163]. With our algorithm we mitigate these sensitivities by learning a representation that works well for detection and by proposing an initialization technique for the weakly trained detectors which proves to avoid many of the pitfalls of prior MIL techniques (see Fig 16.4).

Chapter 15

Detection Adaptation

15.1 Large Scale Detection through Adaptation

We propose a learning algorithm that uses a heterogeneous data source, containing only weak labels for some tasks, to produce strong visual recognition models for all. Our approach is to cast the shift from tasks that require weak labels to tasks that require strong labels as a domain adaptation problem. We then consider transforming the models for the weakly labeled task into the models for the strongly labeled task. For concreteness, we will present our algorithm applied to the specific task shift of classification to detection, called Large Scale Detection through Adaptation (LSDA). In the following section, we will explain how to shift to a different strongly labeled task of semantic segmentation.

Let the set of images with only weak labels be denoted as $\mathcal{W}$ and the set of images with strong labels (bounding box annotations) from auxiliary tasks be denoted as $\mathcal{S}$. We assume that the set of object categories that appear in the weakly labeled set, $\mathcal{C}_\mathcal{W}$, do not overlap with the set of object categories that appear in the strongly labeled set, $\mathcal{C}_\mathcal{S}$. For each image in the weakly labeled set, $I \in \mathcal{W}$, we have an image-level label per category, k: $Y_I^k \in \{1, -1\}$. For each image in the strongly labeled set, $I \in \mathcal{S}$, we have a label per category, k, per region in the image, $i \in R_I$: $y_i^k \in \{1, -1\}$. We seek to learn a representation, $\phi(\cdot)$ that can be used to train detectors for all object categories, $\mathcal{C} = \{\mathcal{C}_\mathcal{W} \cup \mathcal{C}_\mathcal{S}\}$. For a category $k \in \mathcal{C}$, we denote the category specific detection parameter as $\boldsymbol{w}_k$ and compute our final detection scores per region, $\boldsymbol{x}$, as $score_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \phi(\boldsymbol{x})$.

We propose a joint optimization algorithm which learns a feature representation, $\phi(\cdot)$, and detection model parameters, w_k, using the combination of strongly labeled scene-centric data, $\mathcal{S}$, with weakly labeled object and scene-centric data, $\mathcal{W}$. For a fixed representation, one can directly train detectors for all categories represented in the strongly labeled set, $k \in C_\mathcal{S}$. Additionally, for the same fixed representation, we reviewed in the previous section techniques to train detectors for the categories in the weakly labeled data set, $k \in \mathcal{C}_\mathcal{W}$. Our insight is that the knowledge from the strong label set can be used to help guide the optimization for the weak labeled set, and we can explicitly adapt our representation for the

categories of interest and for the generic detection task.

Below, we state our overall objective:

$$\min_{\substack{\boldsymbol{w}_k,\phi \\ k\in\mathcal{C}}} \quad \sum_k \Gamma(\boldsymbol{w}_k) + \alpha_1 \sum_{I\in\mathcal{W}} \sum_{p\in\mathcal{C}_\mathcal{W}} \mathcal{F}(Y_I^p, \boldsymbol{w}_p) + \alpha_2 \sum_{I\in\mathcal{S}} \sum_{i\in R_I} \sum_{q\in\mathcal{C}_\mathcal{S}} \ell(y_i^q, \boldsymbol{w}_q^T \phi(\boldsymbol{x}_i)) \quad (15.1)$$

where $\ell(.)$ is the cross-entropy loss function, $\mathcal{F}$ is the region-based loss function over weak categories, α_1, α_2 are scalar hyper-parameters and $\Gamma(.)$ is a regularization over the detector weights. We use convolutional neural networks (CNNs) to define our representation ϕ and thus the last layer weights serve as detection weights w. We adopt the CNN architecture of [103] (referred to as *AlexNet*).

This formulation is difficult to optimize directly, so we propose to solve this objective by sequentially optimizing easier sub-problems which are less likely to diverge (see Figure 15.1).

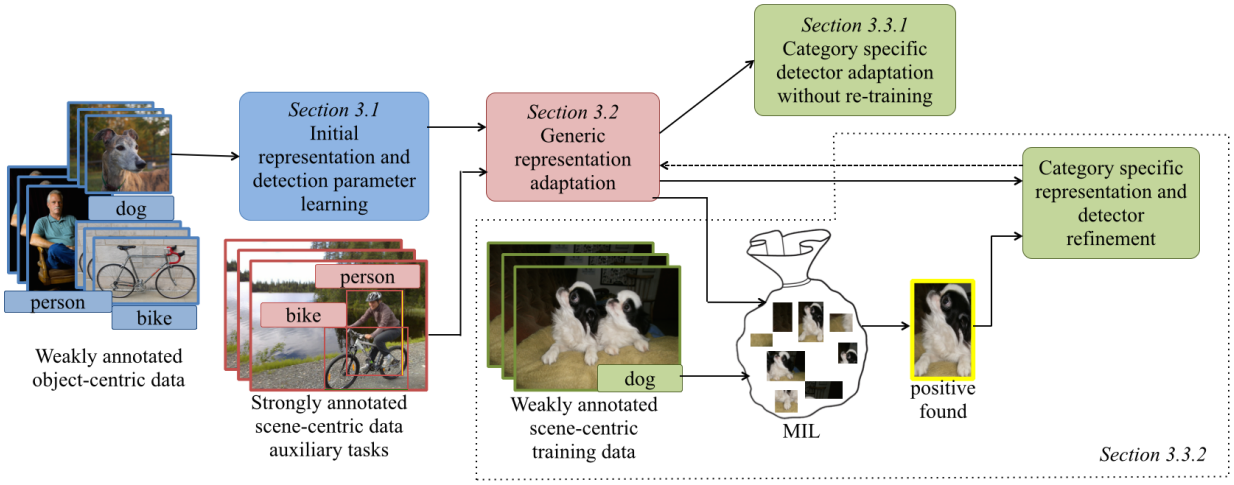


Figure 15.1: Our method (LSDA) jointly optimizes a representation and category specific detection parameters for categories with only weakly labeled data. We first learn a feature representation conducive to adaptation by initializing all parameters with weakly labeled data. We then collectively refine the feature space with strongly labeled data from auxiliary tasks to adapt the category invariant representation from classification to detection (red boxes). Finally, we perform category specific adaptation (green boxes) either without re-training or by solving MIL in our detection feature space and using the discovered bounding boxes to further refine the representation and detection weights.

Lets describe the sub-problems for our overall approach. We begin by initializing a feature representation ϕ and the detection weights $\boldsymbol{w}$ using auxiliary weakly labeled data (Figure 15.1: *blue boxes*). These weights can be used to compute scores per region proposal to produce initial detection scores. We next use available strongly labeled data from auxiliary tasks to transfer category invariant information about the detection problem. We accomplish this through further optimizing our feature representation and learning generic

background detection weights, $\boldsymbol{w}, \phi$, (Figure 15.1: *red boxes*). We then use the well tuned detection feature space to perform MIL on our weakly labeled data to find positive instances (Figure 15.1: *yellow boxes*). Finally, we use our discovered positive instances together with the strongly labeled data from auxiliary tasks to jointly optimize all parameters corresponding to feature representation and detection weights. We now describe each of these steps in detail in the follow subsections.

15.2 Initializing representation and detection parameters

As mentioned earlier, we use the AlexNet architecture to describe representation ϕ and detection weights w. Since this network requires a large amount of data and time to train its approximately 60 million parameters, we start by pre-training on the ILSVRC2012 classification dataset, which we refer to as auxiliary weakly labeled data. It contains 1.2 million weakly labeled images of 1000 categories. Pre-training on this dataset has been shown to be a very effective technique [40, 152, 59], both in terms of performance and in terms of limiting the amount of in-domain labeled data needed to successfully tune the network. This data is usually object centric and is therefore effective for training a network that is able to discriminate between different categories. Next, we replace the last weight layer (1000 linear classifiers) with $K = |\mathcal{C}|$ randomly initialized linear classifiers, one for each category in our task.

We next learn initial values for all of the detection parameters for our particular categories of interest, $\boldsymbol{w}_k,\ \forall k \in \mathcal{C}$. We obtain such initialization by solving the simplified learning problem of image-level classification. The image, $I \in \mathcal{S}$, is labeled as positive for a category k if any of the regions in the image are labeled as positive for k and is labeled as negative otherwise, we denote the image level label as in the weakly labeled case: Y_I^k. Now, we can optimize over all images to refine the representation and learn category specific parameters that can be used per region proposal to produce detection scores:

$$\min_{\substack{\boldsymbol{w}_k,\phi \\ k\in\mathcal{C}}} \sum_k \left[\Gamma(\boldsymbol{w}_k) + \alpha \sum_{I\in\{\mathcal{W}\cup\mathcal{S}\}} \ell(Y_I^k, \boldsymbol{w}_k^T \phi(I)) \right] \tag{15.2}$$

We optimize Equation (15.2) through fine-tuning our CNN architecture with a new K-way last fully connected layer, where $K = |\mathcal{C}|$. This serves as our initialization for solving sequential sub-problems to optimize overall objective (15.1). We find that even using the net trained on weakly labeled data in this way produces a strong baseline. We will refer this baseline as '*Classification Network*' in the experiments; see Table 16.2.

15.3 Net Surgery to Change Classifiers into Detectors

We next transform our classification network into a detection network and learn a representation which makes it possible to separate objects of interest from background and makes it easy to distinguish different object categories. We proceed by modifying the representation (layers 1-7), $\phi(\cdot)$, through finetuning, using the available strongly labeled data for categories in set $\mathcal{C}_\mathcal{S}$. Following the Regions-based CNN (R-CNN) [59] algorithm, we collect positive bounding boxes for each category in set $\mathcal{C}_\mathcal{S}$ as well as a set of background boxes using a region proposal algorithm, such as selective search [178]. We use each labeled region as a fine-tuning input to the CNN after padding and warping it to the CNN's input size. Note that the R-CNN fine-tuning algorithm requires bounding box annotated data for all categories and so can not directly be applied to train all K detectors. Fine-tuning transforms all network weights (except for the linear classifiers for categories in $\mathcal{C}_\mathcal{W}$) and produces a softmax detector for categories in set $\mathcal{C}_\mathcal{S}$, which includes a weight vector for the new background class. We find empirically that fine-tuning induces a generic, category invariant transformation of the classification network into a detection network. That is, even though fine-tuning sees no strongly labeled data for categories in set $\mathcal{C}_\mathcal{W}$, the network transforms in a way that automatically makes the original set $\mathcal{C}_\mathcal{W}$ image classifiers much more effective at detection (see Figure 16.6). Fine-tuning for detection also learns a background weight vector that encodes a generic "background" category, $\boldsymbol{w}_b$. This background model is important for modeling the task shift from image classification, which does not include background distractors, to detection, which is dominated by background patches. This detector explicitly attempts to recognize all data labeled as negative in our bags. Since we initialize this detector with the strongly labeled data, we know precisely which regions correspond to background.

This can be summarized as the following intermediate sub-problem for objective (15.1):

$$\min_{\substack{\boldsymbol{w}_q,\phi \\ q\in\{\mathcal{C}_\mathcal{S},b\}}} \sum_q \left[\Gamma(\boldsymbol{w}_q) + \alpha \sum_{I\in\mathcal{S}} \sum_{i\in R_I} \ell(y_i^q, \boldsymbol{w}_q^T \phi(\boldsymbol{x}_i))\right] \tag{15.3}$$

This is accomplished by fine-tuning our CNN architecture with the strongly labeled data, while keeping the detection weights for the categories with only weakly labeled data fixed. We will call this method as '*LSDA rep only*' in our experiments.

15.4 Adapting category specific representation and detection parameters

Finally, we seek to adapt the category dependent representation and model parameters for the categories in our weakly labeled set, $\mathcal{C}_\mathcal{W}$. We will present two approaches to this problem of learning detection weights for weak categories. Specifically, we aim to update the weakly labeled category specific parameters. Section 15.4 presents a heuristic adaptation approach that requires no further CNN training with gradient descent and updates only the weakly

labeled classification parameters. Section 15.4 describes a separate adaptation approach that directly optimizes a subproblem of our overall objective (15.1). It uses multiple instance learning to discover localized labeled regions in the weakly labeled training data and uses the discovered labels to adapt both the representation and the classification parameters for categories in the weakly labeled set.

K-nearest neighbors based adaptation

In this section, we describe a technique for adapting the category specific parameters of the classifier model into the detector model parameters that are better suited for use with the detection feature representation based on a k-NN heuristic. We will determine a similarity metric between each category in the weakly labeled set, $\mathcal{C}_\mathcal{W}$, to the strongly labeled categories, $\mathcal{C}_\mathcal{S}$.

For simplicity, we separate the category specific output layer (8th layer of the network - $fc8$) of the classification model into two components $fc_\mathcal{S}$ and $fc_\mathcal{W}$, corresponding to model parameters for the categories in the strongly labeled set $\mathcal{C}_\mathcal{S}$ and the weakly labeled set $\mathcal{C}_\mathcal{W}$, respectively. During our generic category adaptation of Section 15.2, we trained a new background prediction layer, fc_b.

For categories in set $C_\mathcal{S}$, adaptation to detectors can be learned directly through fine-tuning the category specific model parameters $fc_\mathcal{S}$. This is equivalent to fixing $fc_\mathcal{S}$ and learning a new layer, zero initialized, $\delta\mathcal{S}$, with equivalent loss to $fc_\mathcal{S}$, and adding together the outputs of $\delta\mathcal{S}$ and $fc_\mathcal{S}$.

Let us define the weights of the output layer of the original classification network as W^c, and the weights of the output layer of the adapted detection network as W^d. We know that for a category $i \in \mathcal{C}_\mathcal{S}$, the final detection weights should be computed as $W^d_i = W^c_i + \delta\mathcal{S}_i$. However, since there is no strongly labeled data for categories in $\mathcal{C}_\mathcal{W}$, we cannot directly learn a corresponding $\delta\mathcal{W}$ layer during fine-tuning. Instead, we can approximate the fine-tuning that would have occurred to $fc_\mathcal{W}$ had strongly labeled data been available. We do this by finding the nearest neighbors categories in set $\mathcal{C}_\mathcal{S}$ for each category in set $\mathcal{C}_\mathcal{W}$ and applying the average change. We assume that there are categories in set $\mathcal{C}_\mathcal{S}$ that are similar to those in set $\mathcal{C}_\mathcal{W}$ and therefore have similar weights and similar gradient descent updates.

Here we define nearest neighbors as those categories with the nearest (minimal Euclidean distance) ℓ_2-normalized fc_8 parameters in the classification network. This corresponds to the classification model being most similar and hence, we assume, the detection model should be most similar. We denote the k^{th} nearest neighbor in set $\mathcal{C}_\mathcal{S}$ of category $j \in \mathcal{C}_\mathcal{W}$ as $N_\mathcal{S}(j,k)$, then we compute the final output detection weights for categories in set $\mathcal{C}_\mathcal{W}$ as:

$$\forall j \in \mathcal{C}_\mathcal{W} : W^d_j \;=\; W^c_j + \frac{1}{k}\sum_{i=1}^{k} \delta\mathcal{S}_{N_\mathcal{S}(j,i)} \tag{15.4}$$

Thus, we adapt the category specific parameters even without bounding boxes for categories in set $\mathcal{C}_\mathcal{W}$. In section 20.4 we experiment with various values of k, including taking the full average: $k = |\mathcal{C}_\mathcal{S}|$. We will now refer to this method as '*LSDA rep+kNN*' in our experiments.

MIL training based adaptation

The previous section provides a technique for adapting the category specific model parameters for the weakly labeled categories without any further CNN training. However, we may want to modify our representation and model parameters by explicitly retraining with the weakly labeled data. To do this, we need to discover localization information from the image-level labels. Therefore, we will begin by solving a multiple instance learning (MIL) problem to discover the portion of each image most likely corresponding to the weak image-level label.

With the representation, ϕ, that has now been directly tuned for detection, we fix the parameter weights, $\phi(\cdot)$ and solve for the regions of interest in each weak labeled image. This corresponds to solving the following objective:

$$\min_{\substack{\boldsymbol{w}_p \\ p\in\{\mathcal{C}_\mathcal{W},b\}}} \quad \sum_p \left[\Gamma(\boldsymbol{w}_p) + \alpha \sum_{I\in\mathcal{W}} \mathcal{F}(Y_I^p, \boldsymbol{w}_p)\right] \tag{15.5}$$

$$\mathcal{F} = \max_{i\in R_I} \boldsymbol{w}_p^T \phi(\boldsymbol{x}_i) \tag{15.6}$$

Note, we can decouple this optimization problem and independently solve for each category in our weakly labeled data set, $p \in \mathcal{C}_\mathcal{W}$. Let's consider a single category p. Our goal is to minimize the loss for category p over images $I \in \mathcal{W}$. We will do this by considering two cases. First, if p is not in the weak label set of an image ($Y_I^p = -1$), then all regions in that image should be considered negative for category p. Second, if $Y_I^p = 1$, then we positively label a region $\boldsymbol{x}_i$ if it has the highest confidence of containing object and negatively label all other regions. We perform the discovery of this top region in two steps. At first, we narrow down the set of candidate bounding boxes using the score, $\boldsymbol{w}_p^T\phi(\boldsymbol{x}_i)$, from our fixed representation and detectors from the previous optimization step. This set is then refined to estimate the most likely region to contain a positive instance in a Latent SVM formulation. The implementation details are discussed section 16.2.

Our final optimization step is to use the discovered bounding boxes from our weak dataset to refine our detectors and feature representation from the previous optimization step. This amounts to the subsequent step for minimization of the joint objective described in Equation (15.1). We collectively utilize the strong labels of images in $\mathcal{S}$ and estimated bounding boxes for the weakly labeled set, $\mathcal{W}$, to optimize for detector weights and feature representation, as follows:

$$\min_{\substack{\boldsymbol{w}_k,\phi \\ k\in\{\mathcal{C},b\}}} \quad \sum_k \left[\Gamma(\boldsymbol{w}_k) + \alpha \sum_{I\in\{\mathcal{W}\cup\mathcal{S}\}} \sum_{i\in R_I} \ell(y_i^k, \boldsymbol{w}_k^T\phi(\boldsymbol{x}_i))\right] \tag{15.7}$$

This is achieved by re-finetuning the CNN architecture. This final method is referred to as '*LSDA rep+joint ft*' in our experiments.

Thus, the overall non-convex objective (15.1) is first approximated through initialization in (15.2). This initialization is then used to solve the sequential optimization problems

defined in (15.3) and (15.5). Further, we present two ways to solve (15.5): k-NN based heuristic approach in (15.4) and MIL-based re-training approach in (15.6).

The sub-problem defined in (15.3) decreases the loss for strongly labeled categories and (15.7) decreases the loss for both weak-strong categories. Thus, this ensures that the overall objective (15.1) decreases. The refined detector weights and representation can be used to discover the bounding box annotations for weakly labeled data again, and this process can be iterated over (see Figure 15.1). We discuss re-training strategies and evaluate the contribution of this final optimization step in Section 16.2.

15.5 Detection with LSDA models

We now describe how our adapted network is used for detection at test time (depicted in Figure 15.2). For each test image we extract region proposals and generate $K+1$ scores per region (similar to the R-CNN [59] pipeline), one score for each category and an additional score for the background category. The score is generated by passing the properly warped image patch through our adapted representation layers and then through one of our proposed category specific adapted layers (described in the previous sections). Finally, for a given region, the score for category i is computed by linearly combining the per category score with the background score: $score_i - score_{\text{bg}}$.

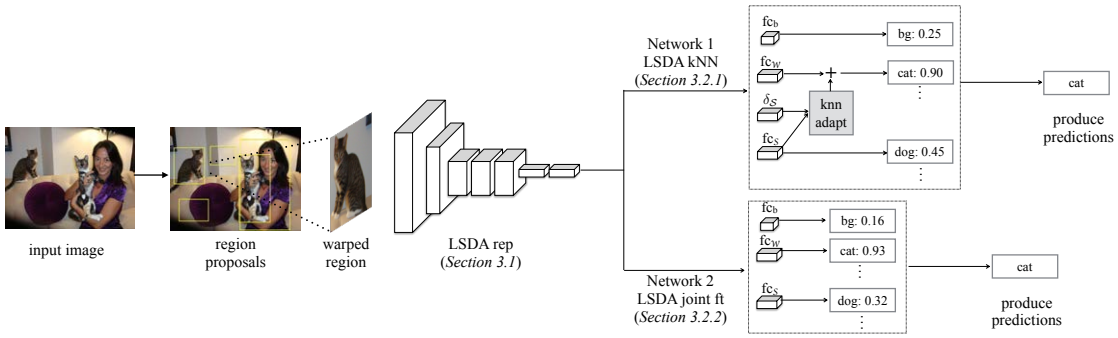


Figure 15.2: Detection with the LSDA network (test time). Given an image, extract region proposals, reshape the regions to fit into the network size and pass through our adapted network. Use the adapted representation and the category specific adaptation either through the no retraining nearest neighbor method or by retraining with our MIL based method. Finally produce detection scores per category for the region by considering background and category scores.

In contrast to the R-CNN [59] model which trains SVMs on the extracted features from layer 7 and bounding box regression on the extracted features from layer 5, we directly use the final score vector to produce the prediction scores without either of the retraining steps. This choice results in a small performance loss, but offers the flexibility of being able to

directly combine the classification portion of the network that has no detection labeled data, and reduces the training time from 3 days to roughly 5.5 hours.

15.6 Recognition Beyond Detection

In the previous section we outline an algorithm for producing weakly supervised detection models which label and coarsely localize objects in scene-centric images. While a bounding box around an object offers significantly more information than an image-level label, it is not sufficiently localized for tasks such as robotic manipulation and full scene parsing. Instead, we would like to produce semantic segmentation models which are capable of labeling each pixel in an image with the object category or background label.

Prior work has shown that convolutional networks can also be applied to arbitrary-sized inputs to allow for per-pixel spatial output. For example, [130] augmented the LeNet digit classification model [111], enabling recognition of strings of digits, and [185] use networks to output 2-dimensional maps in order to identify the locations of postal address blocks. This technique has been used to produce semantic segmentation outputs of *C. elegans* [135] and more recently for generic object categories [123]. These "fully convolutional" networks can also be finetuned end-to-end on segmentation ground truth to produce fully supervised segmentation models [123].

As we would like to produce pixel level labels from our LSDA model, we will build off of our recent work for object category semantic segmentation [123]. However, [123] requires full semantic segmentation (pixel-level) annotations to train the corresponding fully connected network. This form of supervision is particularly expensive to collect and in general very few data sources exist with these annotations.

Instead, we argue that much of the knowledge gained through training with pixel-level annotations can be transferred from the much weaker bounding box annotations. Therefore, we demonstrate that a reasonable semantic segmentation is possible by directly using detection parameters in a fully convolutional framework. Further, we show that even our weakly supervised detection models presented in the previous section are able to localize objects more precisely than a bounding box, despite never receiving pixel-level annotations and for many categories never even receiving bounding box annotations.

To produce such a network we take our final adapted LSDA model, which for the purpose of our experiments was trained using an AlexNet basic architecture [103], and we convert the model into the corresponding fully convolutional 32 stride network (FCN-32s) presented by [123]. This amounts to relatively few changes to the network architecture. First, each input image is padded with 100 pixels before features are extracted. Next each of the three fully connected layers are converted into convolutional layers, where layer 6 has 4096 convolutions with 6×6 sized kernels, layer 7 has 4096 convolutions with 1×1 sized kernels, and the final score layer has $K+1$ convolutions with 1×1 sized kernels (where K is the number of categories, plus one for background). Finally, additional deconvolution and crop layers are added which up-sample the score map produced by the 8th layer (bilinear interpolation)

and crops the pixel level score map to be the size of the input image. This means the final output of the network is a score per category per pixel, which allows us to perform semantic segmentation.

Chapter 16

Experiments

To demonstrate the effectiveness of our approach we present quantitative results on the ILSVRC2013 detection dataset. The dataset offers images exhaustively labeled with bounding box annotations for 200 relevant object categories. The training set has ∼400K labeled images and on average 1.534 object classes per image. The validation set has 20K labeled images with ∼50K labeled objects. We simulate having access to weak labels for all 200 categories and having strong labels for only the first 100 categories (alphabetically sorted).

16.1 Experiment Setup & Implementation Details

We start by separating our data into classification and detection sets for training and a validation set for testing. Since the ILSVRC2013 training set has on average fewer objects per image than the validation set, we use this data as our classification data. To balance the categories we use ≈1000 images per class (200,000 total images). **Note**: for classification data we only have access to a single image-level annotation that gives a category label. In effect, since the training set may contain multiple objects, this single full-image label is a weak label, even compared to other classification training data sets. Next, we split the ILSVRC2013 validation set in half as [59] did, producing two sets: val1 and val2. To construct our detection training set, we take the images with bounding box labels from val1 for only the first 100 categories (≈ 5000 images). Since the validation set is relatively small, we augment our detection set with 1000 bounding box labeled images per category from the ILSVRC2013 training set (following the protocol of [59]). Finally we use the second half of the ILSVRC2013 validation set (val2) for our evaluation.

We implemented our CNN architectures and execute all fine-tuning using the open source software package Caffe [92] and have made our model definitions weights publicly available.

We use the ILSVRC13 detection dataset [35] for our experiments. This dataset provides bounding box annotations for 200 categories. The dataset is separated into three pieces: train, val, test (see Table 16.1). The training images have fewer objects per image on an average than validation set images, so they constitute classification style data [85]. Following

Train	Num images	395905
	Num objects	345854
Val	Num images	20121
	Num objects	55502

Table 16.1: Statistics of the ILSVRC13 detection dataset. Training set has fewer objects per image than validation set.

Layers Adapted using Strongly Labeled Data	mAP (%) Weak Categories	mAP (%) All Categories
No Adapt (Classification Network)	10.31	11.90
fc_{bgrnd}	12.22	13.60
fc_{bgrnd}, fc_6	13.72	19.20
fc_{bgrnd}, fc_7	14.57	19.00
fc_{bgrnd}, $\text{fc}_{\mathcal{S}}$	11.74	14.90
fc_{bgrnd}, fc_6, fc_7	14.20	20.00
fc_{bgrnd}, fc_6, fc_7, $\text{fc}_{\mathcal{S}}$	14.42	20.40
fc_{bgrnd}, layers 1-7, $\text{fc}_{\mathcal{S}}$	**15.85**	**21.83**

Table 16.2: Ablation study for different techniques for category independent adaptation of our model (LSDA rep only). We consider training with the first 100 (alphabetically) categories of the ILSVRC2013 detection validation set (on val1) and report mean average precision (mAP) over the 100 weakly labeled categories (on val2). We find the best improvement is from fine-tuning all layers.

prior work [59], we use the further separation of the validation set into val1 and val2. Overall, we use the train and val1 set for our training data source and evaluate our performance of the data in val2.

16.2 Quantitative Analysis of Adapted Representation

We evaluate the importance of each component of our algorithm through an ablation study. As a baseline, we consider training the network with only the weakly labeled data (no adaptation) and applying the network to the region proposals.

In Table 16.2, we present a detailed analysis of the different category independent adaptation techniques we could use to train the network. We call this method LSDA rep only. We find that the best category invariant adaptation approach is to learn the background

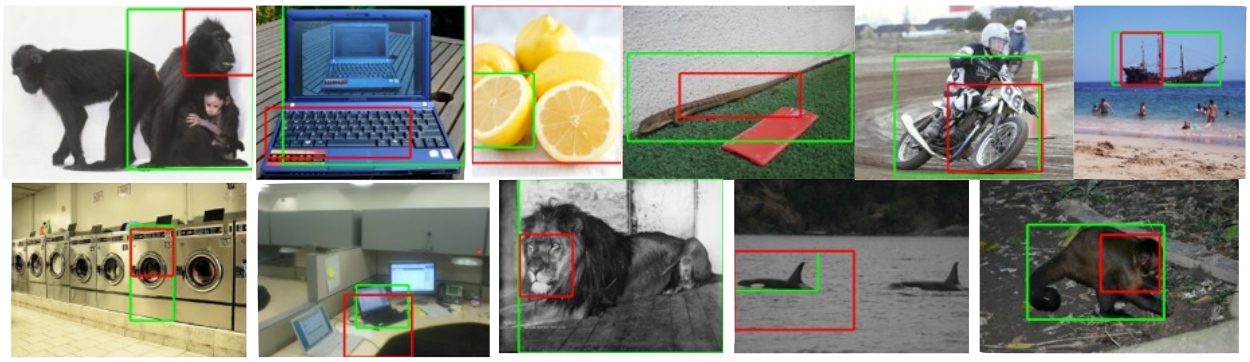


Figure 16.1: We show example detections on weakly labeled categories, for which we have **no detection training data**, where LSDA (shown with green box) correctly localizes and labels the object of interest, while the classification network baseline (shown in red) incorrectly localizes the object. This demonstrates that our algorithm learns to adapt the classifier into a detector which is sensitive to localization and background rejection.

category layer and adapt all convolutional and fully connected layers, bringing mAP on the weakly labeled categories from 10.31% up to 15.85% i.e. this achieves a 54% relative mAP boost over the classification only network. We later observe that the most important step of our algorithm proved to be adapting the feature representation, while the least important was adapting the category specific parameter. This fits with our intuition that the main benefit of our approach is to transfer category invariant information from categories with known bounding box annotation to those without the bounding box annotations.

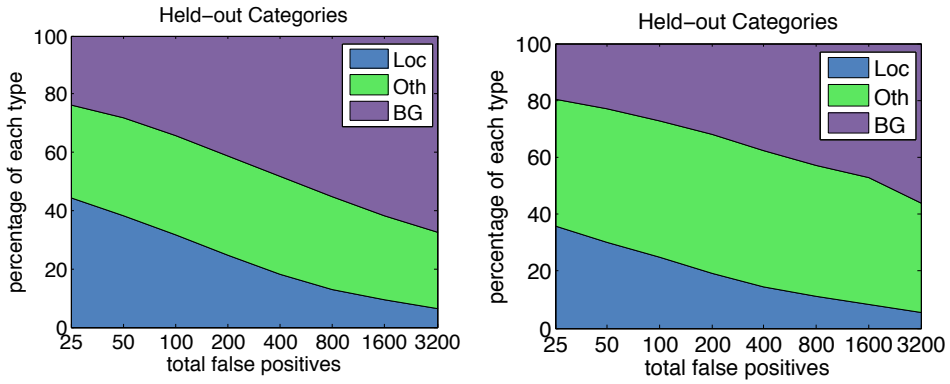
We find that one of the biggest reasons our algorithm improves is from reducing localization error. For example, in Figure 16.1, we show that while the classification only trained net tends to focus on the most discriminative part of an object (ex: face of an animal) after our adaptation, we learn to localize the whole object (ex: entire body of the animal).

Error Analysis on Weakly Labeled Categories

We next present an analysis of the types of errors that our system (LSDA) makes on the weakly labeled object categories. First, in Figure 19.3, we consider three types of false positive errors: Loc (localization errors), BG (confusion with background), and Oth (other error types, which is essentially correctly localizing an object, but misclassifying it). After separating all false positives into one of these three error types we visually show the percentage of errors found in each type as you look at the top scoring 25-3200 false positives.[1] We consider the baseline of starting with the classification only network and show the false positive breakdown in Figure 16.2a. Note that the majority of false positive errors are confusion with background and localization errors. In contrast, after adapting the network using LSDA

[1]We modified the analysis software made available by [75] to work on ILSVRC-2013 detection

we find that the errors found in the top false positives are far less due to localization and background confusion (see Figure 16.2b). Arguably one of the biggest differences between classification and detection is the ability to accurately localize objects and reject background. Therefore, we show that our method successfully adapts the classification parameters to be more suitable for detection.



(a) Classifier Network (b) LSDA rep+knn

Figure 16.2: Comparison of error type breakdown on the categories which have no training bounding boxes available (weakly labeled data). After adapting all the layers in the network (LSDA), the percentage of false positive errors due to localization and background confusion is reduced (b) as compared to directly using the classification network for detection (a).

In Figure 16.3, we show examples of the top scoring *O*th error types for LSDA on the weakly labeled data. This means the detector localizes an incorrect object type. For example, the motorcycle detector localized and mislabeled bicycle and the lemon detector localized and mislabeled an orange. In general, we noticed that many of the top false positives from the *O*th error type were confusion with very similar categories. This is discussed in detail in next subsection.

Figure 16.3: Examples of the top scoring false positives from our LSDA rep+knn network. Many of our top scoring false positives come from confusion with other categories.

Analysis of Discovered Boxes

We now analyze the quality of boxes discovered using adaptation of all layers including the background class. One of the key components of our system is using strong labels from auxiliary tasks to learn a representation where it's possible to discover bounding boxes that correspond to the objects of interest in our weakly labeled data source. We begin our analysis by studying the bounding box discovery that our feature space enables, using selective search [178] to produce candidate regions. We optimize the bounding box discovery (Equations (15.5),(15.6)) using a one vs all Latent SVM formulation and optimize the formulation for AUC criterion [17]. This ensures that the top candidate regions chosen for joint fine-tuning have high precision. The feature descriptor used is the output of the fully connected layer, fc_7, of the CNN which is produced after fine-tuning the feature representation with strongly labeled data from auxiliary tasks. Following our alternating minimization approach, these discovered top boxes are then used to re-estimate the weights and feature representations of our CNN architecture.

	CorLoc over full dataset				Localization mAP (%)
	ov=0.3	ov=0.5	ov=0.7	ov=0.9	ov=0.5
Classification Network	29.63	26.10	24.28	23.43	13.13
LSDA rep only	32.69	28.81	26.27	24.78	22.81

Table 16.3: CorLoc over dataset and localization mAP (i.e. given the labels) performance of discovered bounding boxes in our weakly labeled training set (val1) of ILSVRC13 detection dataset. Comparison with varying amount of overlap with ground truth box. About 25% of our discovered boxes have an overlap of at least 0.9. Our method is able to significantly improve the quality of discovered boxes after incorporating strong labels from auxiliary tasks.

To evaluate the quality of discovered boxes, we do ablation study analyzing their overlap with ground truth which is measured using the standard intersection over union (IOU) metric. Table 16.3 reports the CorLoc for varying overlapping thresholds. CorLoc across full dataset is defined as the accuracy of discovered boxes i.e. the accuracy that the box is correctly localized per image at different thresholds. Our optimization approach produces one positive bounding box per image with a weak label, and a discovered box is considered a true positive if it overlaps sufficiently with the ground truth box that corresponds to that label. Since each bounding box, once discovered, is considered an equivalent positive (regardless of score) for the purpose of retraining the '*LSDA rep only*' model, this simple CorLoc metric is a good indication of the usefulness of our discovered bounding boxes. We note here that after re-training with our mined boxes the CorLoc will further improved, as indicated in the detection mAP reported in the next section.[**I**]t is interesting that a significant fraction of discovered boxes have high overlap with the ground truth regions. For reference, we also computed the standard mean average precision over the discovered boxes for localization task i.e. when label is known. It is important to note that the improvement

in localization mAP is much more significant than the CorLoc. This is because mAP is obtained by averaging over recall values, and the '*LSDA rep only*' model achieves better overall recall than the '*Classification Network*' model.

It is important to understand not only that our new feature space improves the quality of the resulting bounding boxes, but also what type of errors our method reduces. In Figure 16.4, we show the top 5 scoring discovered bounding boxes before and after modifying the feature space with strong labels from auxiliary tasks. We find that in many cases the improvement comes from better localization. For example without auxiliary strong labels we mostly discover the face of a lion rather than the body that we discover after our algorithm. Interestingly, there is also an issue with co-occurring classes. We are better able to localize "lion" body rather than the face. Most amazing results are for the "ping-pong" and "rugby" (second and third row) category where we are actually able to mine boxes for the racket and ball, while the classification net could only get the person boxes which is incorrect. Once we incorporate strong labels from auxiliary tasks we begin to be able to distinguish the person playing from the racket/ball itself. In the bottom row of Figure 16.4, we show the top 5 discovered bounding boxes for "tennis racket" where we are partially able to correct the images. Finally, there are some example discovered bounding boxes where we reduce quality after incorporating the strong labels from auxiliary tasks. For example, one of our strongly labeled categories is "computer keyboard". Due to the strong training with keyboard images, some of our discovered boxes for "laptop" start to have higher scores on the keyboard rather than the whole laptop (see Figure 16.5). Also for the "water-craft" category, our adapted network ignores the mast but better localizes the boat itself. which slightly decreases the IOU of obtained box.

Detection Performance on ILSVRC13

Now that we have analyzed the intermediate result of our algorithm, we next study the full performance of our system. Figure 16.6 shows the mean average precision (mAP) percentage computed over the categories in val2 of ILSVRC13 for which we only have weakly labeled training data (categories 101-200). Previous method, LCL [183], detects in the standard weakly supervised setting – having no bounding box annotations for any of the 200 categories. This method also only reports results across all 200 categories on the full validation set. Our experiments indicate that the first 100 categories are easier on average then the second 100 categories, therefore the 6.0% mAP may actually be an upper bound of the performance of this approach. We also compare our algorithm against the scenario when the class-specific layer is adapted using nearest neighbors across all categories (LSDA rep+knn). The joint representation and multiple instance learning approach achieves the highest results (LSDA rep+joint ft).

We next consider different re-training strategies for learning new features and detection weights after discovering the bounding boxes in the weakly labeled data. Table 16.4 reports the mean average precision (mAP) percentage for no re-training (directly using the feature space learned after incorporating the strong labels), LSDA rep only, no retraining

Figure 16.4: Example discovered bounding boxes learned using our method. Left side shows the discovered boxes after fine-tuning with images in classification settings only, and right side shows the discovered boxes after fine-tuning with auxiliary strongly labeled dataset. We show top 5 discovered boxes across the dataset for corresponding category. Examples with a green outline are categories for which our algorithm was able to correctly discover bounding boxes of the object, while the feature space with only weak label training was not able to produce correct boxes. After incorporating the strong labels from auxiliary tasks, our method starts discovering "ping-pong" racket/ball and "rugby" ball, though still has some confusion with the person playing tennis. None of the discovered boxes from the original feature space correctly located racket/ball and instead included the person as well. In yellow we highlight the specific example of "tennis racket", where some of the boxes get corrected not all top boxes.

Figure 16.5: Example discovered boxes of the category "laptop" where using auxiliary strongly labeled data causes bounding box discovery to diverge. *Left*: The discovered boxes obtained after fine-tuning with images in classification settings only. *Right*: The discovered boxes obtained after fine-tuning with the auxiliary strongly labeled dataset that contains the category "computer keyboard". These boxes were low scoring examples, but we show them here to demonstrate a potential failure case – specifically, when one of the strongly labeled classes is a part of one of the weakly labeled classes. In the second example, adapted network better localizes the "water-craft" but misses the mast which decreases the IOU slightly.

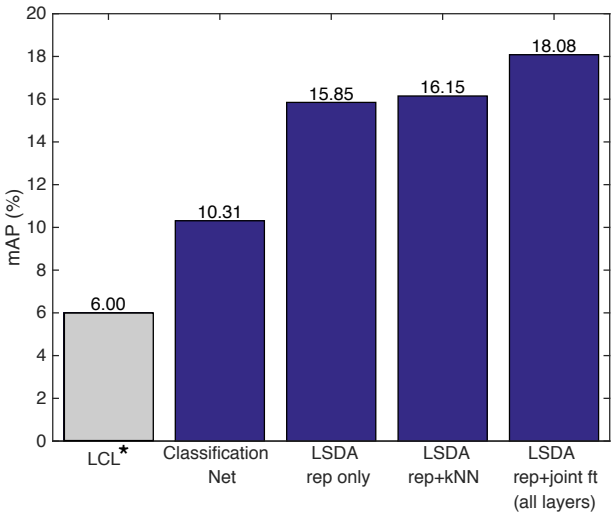


Figure 16.6: Comparison of mAP (%) for categories without any bounding box annotations (101-200 of val2) of ILSVRC13. The Joint representation and category-specific learning using MIL outperforms all other approaches. *As a reference we report the performance of LCL [183] which was computed across all 200 categories of the full validation set (val1+val2).

Category Specific Adaptation Strategy	mAP (%) Weak Categories	mAP (%) All Categories
LSDA rep only	15.85	21.83
LSDA rep+kNN (k=5)	15.97	22.05
LSDA rep+kNN (k=10)	16.15	22.05
LSDA rep+kNN (k=100=$\|\text{fc}_{\mathcal{S}}\|$)	15.96	21.94
LSDA rep+joint ft ($\text{fc}_{\mathcal{W}}$)	17.01	22.43
LSDA rep+joint ft (all layers)	**18.08**	**22.74**
Baseline: Classification Network	10.31	11.90
Oracle: RCNN Full Detection Network	26.25	28.00

Table 16.4: Comparison of different ways to re-train after discovery of bounding boxes. We show mAP on val2 set from ILSVRC13. We find that the most effective way to re-train with discovered boxes is to modify the detectors and the feature representation.

but last layer weights of weak categories adapted using nearest neighbors, LSDA rep+knn, re-training only the category-specific detection parameters, LSDA rep+joint ft ($\text{fc}_{\mathcal{W}}$), and retraining feature representations jointly with category-specific weights, LSDA rep+joint ft (all layers). In our experiments the improved performance is due to the first iteration of the overall algorithm. We find that the best approach is to jointly learn to refine the feature representation and the category-specific detection weights. More specifically, we learn a new feature representation by fine-tuning all fully connected layers in the CNN architecture. The last row shows the performance achievable by our detection network if it had access to bounding box annotated data for all 200 categories, and serves as a performance upper bound.[2] Our method achieves **18.08%** mAP on weakly labeled categories as compared to **10.31%** of baseline, but it is still significantly lower than fully-supervised oracle which gives **26.25%**.

We finally analyze examples where our full algorithm which jointly learns representation and class-specific layer using MIL (LSDA rep+joint ft) outperforms the previous approach where only representation is adapted without joint learning over weak labels (LSDA rep+knn). Figure 16.7 shows a sample of the types of errors our algorithm improves on. These include localization errors, confusion with other categories, and interestingly, confusion with co-occurring categories. In particular, our algorithm provides improvement when searching for a small object (ball or helmet) in a sports scene. Training only with weak labels causes the previous state-of-the-art to confuse the player and the object, resulting in a detection that includes both. Our algorithm is able to localize only the small object and

[2]To achieve R-CNN performance requires additionally learning SVMs on the activations of layer 7 and bounding box regression on the activations of layer 5. Each of these steps adds between 1-2mAP at high computation cost and using the SVMs removes the adaptation capacity of the system.

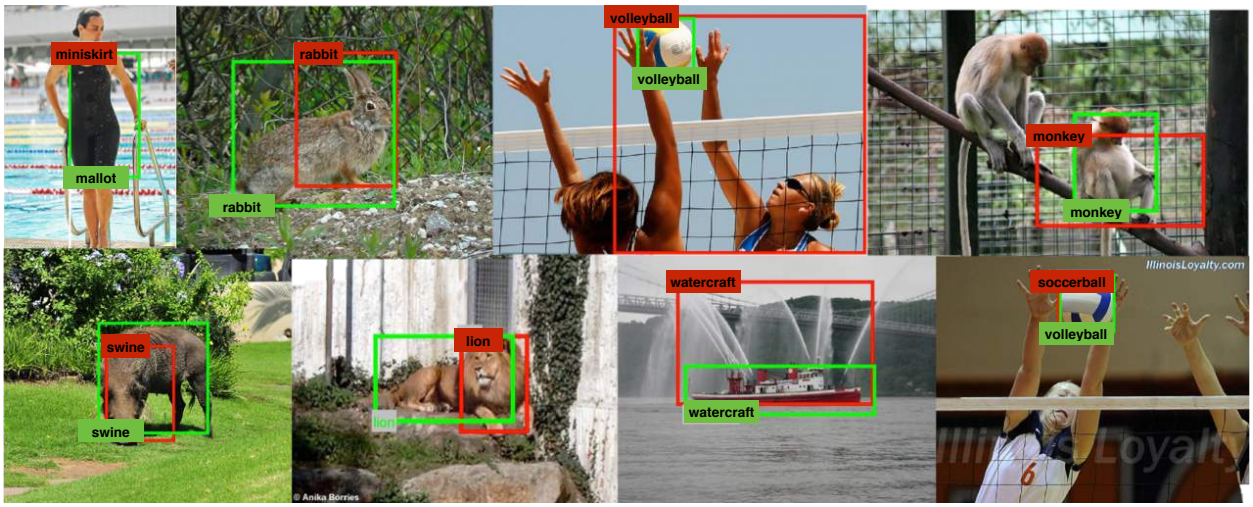


Figure 16.7: Examples where our algorithm after joint MIL adaptation (LSDA rep+joint ft) outperforms the representation only adaptation (LSDA rep only). We show the top scoring detection from LSDA rep only with a Red box and label, and the top scoring detection from LSDA rep+joint ft, as a Green box and label. Our algorithm improves localization (ex: rabbit, lion etc), confusion with other categories (ex: miniskirt vs maillot), and confusion with co-occurring classes (ex: volleyball vs volleyball player)

recognize that the player is a separate object of interest.

16.3 Large Scale Detection

To showcase the capabilities of our technique we produced a 7604 category detector. The first categories correspond to the 200 categories from the ILSVRC2013 challenge dataset which have bounding box labeled data available. The other 7404 categories correspond to leaf nodes in the ImageNet database and are trained using the available full image weakly labeled classification data. We trained a full detection network using the 200 strongly labeled categories and trained the other 7404 last layer nodes using only the weak labels. Note, the ImageNet dataset does contain other non-exhausitvely labeled images for around 3000 object categories, 1825 of which overlap with the 7404 leaf node categories in our model. We do not use these labels during training of our large scale model. Quantitative evaluation for these categories is difficult to compute since they are not exhaustively labeled, however a followup work by [132] evaluated F1 score of our model for the few object instances labeled per image to be 9.59%. Also note that while we have no bounding box annotations for the 7404 fine-grained categories, some may be related to the 200 basic level categories for which we use bounding box data to train – for example a particular breed of dog from 7404 weakly labeled data while 'dog' appears in the 200 strongly labeled categories.

Figure 16.8: Example top detections from our 7604 category detector. Detections from the 200 categories that have bounding box training data available are shown in blue. Detections from the remaining 7404 categories for which only weakly labeled data is available are shown in red.

We show qualitative results of our large scale detector by displaying the top detections per image in Figure 16.8. The results are filtered using non-max suppression across categories to only show the highest scoring categories.

The main contribution of our algorithm is the joint representation and multiple instance learning approach for modifying a convolutional neural network for detection. However, the choice of network and how the net is used at test time both effect the detection time computation. We have therefore also implemented and released a version of our algorithm running with fast region proposals [102] on a spatial pyramid pooling network [72], reducing our detection time down to half a second per image (from 4s per image) with nearly the same performance. We hope that this will allow the use of our 7.6K model on large data sources such as videos. We have released the 7.6K model and code to run detection (both the way presented in this paper and our faster version) at `lsda.berkeleyvision.org`.

16.4 Fully Convolutional LSDA for Semantic Segmentation

Bounding boxes localize objects to an inherently limited degree. While the system presented so far produces remarkably accurate bounding boxes from weak training labels, it does not address the ultimate goal of knowing exactly which pixels correspond to which objects.

Segmentation ground truth is unavailable for all but a few of the 7604 categories in our large scale detector, and segmentations are even more costly to annotate than bounding boxes. Nevertheless, as described in Section 16.4, we can convert our detection-adapted network into a fully-convolutional model following [123] and produce dense outputs for each of the 7604 categories plus 1 for background. We call this model LSDA7k FCN-32s since we use the 32 stride version of the fully convolutional networks proposed in [123]. We next evaluate our semantic segmentation model using the PASCAL dataset [49] and the following

metrics.

Metrics We compute both the commonly used mean intersection over union (mean IU) metric for semantic segmentation as well as three other metrics used by [123]. The metrics are defined below, where n_{ij} denotes the number of pixels from class i predicted to belong to class j so that the number of pixels belonging to class i are $m_i = \sum_j n_{ij}$, and K denotes the number of classes.

- pixel accuracy: $\sum_i n_{ii} / \sum_i m_i$
- mean accuracy: $1/K \sum_i n_{ii}/t_i$
- mean IU: $1/K \sum_i n_{ii}/(m_i + \sum_j n_{ji} - n_{ii})$
- frequency weighted IU: $(\sum_l m_l)^{-1} \sum_i m_i n_{ii}/(m_i + \sum_j n_{ji} - n_{ii})$

We would like to understand how well our model can localize weakly trained objects so for each of the PASCAL 20 object categories we manually find the set of fine-grained categories from the 7404 weakly labeled leaf nodes in ImageNet that correspond to that category. Since layer 8 of our LSDA7k FCN-32s network produces 7605 outputs per region of the image, we insert an additional mapping layer which for each category c is the maximum score across all weakly labeled categories which correspond to that PASCAL category. Next, this reduced score map where each image region now has 21 scores is run through the deconvolution layer to produce the corresponding PASCAL per pixel scores. Finally, for each pixel we choose a label based on which of the categories or background has the highest pixel score.

We report results on both the PASCAL 2011 and 2012 validation sets. Note, our method was not trained on any PASCAL images and in general was trained for classification of 7404 fine-grained categories and then adapted using our algorithm for detection. Additionally, our model is trained using the AlexNet architecture while most state-of-the-art semantic segmentation models are trained using the larger VGG network [159].

For the PASCAL 2011 validation set, shown in Table 16.5, we first compare against the classification model trained for the 7404 category full image labels. We run this model fully convolutionally using the FCN-32s approach (AlexNet) and report the segmentation performance in the first row as *Classification 7K FCN-32s (AlexNet)*. This method gives a baseline for our LSDA approach which uses this model as the initialization prior to our adaptation approach. Next, we compare against the reported performance of the weakly trained models of [142] and for reference, the fully supervised AlexNet and VGG FCN-32s presented by [123]. We report all four metrics for our work and report all available metrics for competing works. We see that our weakly trained model outperforms the baseline classification model run fully convolutionally and almost reaches the performance of the MIL-FCN method which uses the higher capacity VGG model and trains specifically for the segmentation task.

Method	pixel acc	mean acc	mean IU	f.w. IU
Classification 7k FCN-32s (AlexNet)	13.3	43.2	11.6	6.7
MIL-FCN (VGG) [142]	-	-	25.0	-
LSDA7k FCN-32s (AlexNet)	70.6	35.5	21.3	59.2
FCN-32s (supervised AlexNet)[123]	85.8	61.7	48.0	76.5
FCN-32s (supervised VGG)[123]	89.1	73.3	59.4	81.4

Table 16.5: **Semantic Segmentation Results for PASCAL 2011 validation set.**

Method	bgrnd	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
EM Adapt (VGG) [140]	65.0	27.9	17.0	26.5	21.2	29.2	48.0	44.8	43.8	15.0	33.8	25.0	39.9	34.0	41.3	31.8	22.9	35.2	23.2	39.3	30.4	33.1
CCNN (VGG) [141]	65.9	23.8	17.6	22.8	19.4	36.2	47.3	46.9	47.0	16.3	36.1	22.2	43.2	33.7	44.9	39.8	29.9	33.4	22.2	38.8	36.3	34.5
LSDA7k FCN-32s (AlexNet)	74.6	17.1	16.1	9.6	7.7	18.5	10.4	27.3	20.8	7.3	9.9	5.5	19.0	12.7	8.5	19.3	14.8	15.2	12.7	20.5	15.2	17.3

Table 16.6: **Semantic Segmentation Results (mean IU%) for PASCAL 2012 validation set.**

The per-category results of our method on the PASCAL 2012 validation set as compared to two state-of-the-art weakly trained semantic segmentation models is shown in Table 16.6. Not surprisingly, our LSDA7k FCN-32s underperforms these methods. No doubt adding the multiple instance loss of [142] or the object constraints of [141], while training directly on the PASCAL dataset would further improve our method. The purpose of these experiments is to give the reader an accurate picture of how well our large scale model performs at pixel level annotation without any tuning to the new situation.

We next show qualitative segmentation segmentation results across the fine-grained 7404 categories of our LSDA7k FCN-32s network in Figure 16.9[3] and compare against the baseline Classification 7K FCN-32s network. We find that often the segmentation masks from our LSDA network are more precise (see "American egret" example) and the top scoring predicted class is often more accurately labeled. For example, the bottom image is labeled as "air conditioner" by the classification network and correctly as "venetian blind" by our network. These category models were trained without ever seeing any associated pixel-level annotations and only potentially see bounding box annotations for related classes (ex: a "dog" bounding box may be used in training but not a "dalmation"). We expect that further adaptation with a multiple instance loss or given a small amount of pixel-level semantic segmentation training data would further refine our models producing tigher object localization.

[3]The full network without the mapping layer to pascal 20 categories.

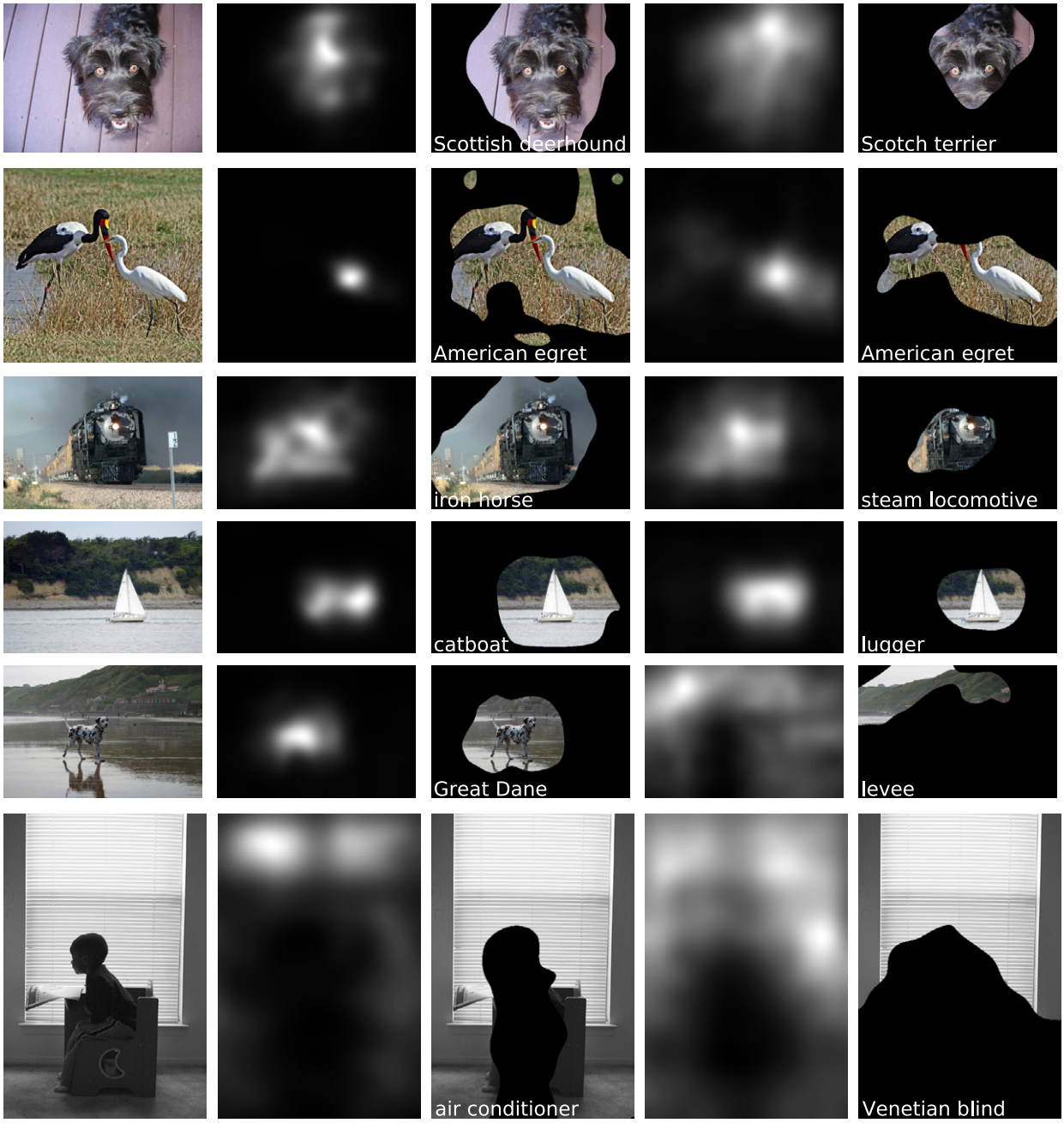


Figure 16.9: We show here qualitative semantic segmenation results comparing our LSDA7k FCN-32s network with the baseline Classification 7k FCN-32s network. Each row shows (from left to right) a test image, predicted heatmap of top scoring class from the classification network, rough segmentation from the classification network, predicted heatmap of the top scoring class from our LSDA network, and the corresponding rough segmentation from the LSDA network. Each segment mask is obtained using a single fixed threshold across all classes ($e^{5/6}$) and for both methods. These examples are selected to illustrate segment quality when the predicted label is reasonable. Although segment quality is far from perfect, it is impressive given that only full-image ground truth labels were available for these categories.

Chapter 17

Summary

We have presented an algorithm that is capable of transforming a classifier into a detector. Our multi-stage algorithm uses corresponding weakly labeled (image-level annotated) and strongly labeled (bounding box annotated) data to learn the change from a classification CNN network to a detection CNN network, and applies that difference to future classifiers for which there is no available strongly labeled data. We then further demonstrate that our adapted detection models can be run fully convolutionally to produce a semantic segmentation model.

Our method jointly trains a feature representation and detectors for categories with only weakly labeled data. We use the insight that strongly labeled data from auxiliary tasks can be used to train a feature representation that is conducive to discovering bounding boxes in weakly labeled data. We demonstrate using a standard detection dataset (ILSVRC13 detection) that our method of incorporating the strongly labeled data from auxiliary tasks is very effective at improving the quality of the discovered bounding boxes. We then use all strong labels along with our discovered bounding boxes to further refine our feature representation and produce our final detectors. We show that our full detection algorithm significantly outperforms both the previous state-of-the-art methods which uses only weakly labeled data, as well as the algorithm which uses strongly labeled data from auxiliary tasks, but does not incorporate any MIL for the weak tasks.

We show quantitatively that without seeing any bounding box annotated data, we can increase performance of a classification network by 50% relative improvement using our adaptation algorithm. Given the significant improvement on the weakly labeled categories, our algorithm enables detection of tens of thousands of categories. We produce a 7.6K category detector and have released both code and models at `lsda.berkeleyvision.org`.

Our approach significantly reduces the overhead of producing a high quality detector. We hope that in doing so we will be able to minimize the gap between having strong large-scale classifiers and strong large-scale detectors. Further we show that large-scale detectors can be used to produce large-scale semantic segmenters. We present semantic segmentation performance for the large scale model on PASCAL VOC with a manual mapping from the 7404 weakly labeled object categories to the 20 categories in the PASCAL dataset. For future work we would like to experiment with incorporating some pixel-level annotations for

a few object categories. Our intuition is that by doing so we will be able to further improve our large-scale models with minimal extra supervision.

Part VI

Adapting Across Visual Modalities

Chapter 18

Depth Modality

18.1 Introduction

RGB and depth images offer different and often complementary information. In fact, recent work has shown that the two image modalities can be used simultaneously to produce better recognition models than either modality alone [71, 183, 182].

Accurate object detection is an essential component for many robotic tasks like mapping, motion planning, grasping and object manipulation. This has motivated the use of depth information from commodity RGB-D sensors to improve object recognition performance [70, 109, 108, 171]. Similarly, object detection is crucial for other commercial applications like surveillance and self-driving cars for which a depth sensor may not always be available. When depth sensory information is available at both training and test time, we refer the reader to previously proposed fully supervised algorithms.

In this part, we focus our study on adaptation across the RGB and depth modality representations. In particular, we consider two different and relevant problem statements. In Chapter 19 we consider the scenario where annotated depth information is missing for the categories we would like to detect at test time and so we present an algorithm to learn a generic, category agnostic depth representation which may be used together with a category specific RGB representation at test time. This problem statement is particularly useful to produce a large scale object detector which may utilize depth information at test time without requiring fully supervised training.

In Chapter 20, we consider the scenario where we use available annotated depth training images as side information by which we may improve a test time RGB detector. We learn an additional RGB representation which hallucinates mid-level depth representations, thereby encoding some of the complementary information that would be extracted from the depth image. This approach can be immediately applied to any RGB model to improve the test time performance when an RGB only camera is available during deployment.

18.2 Prior Work

We review three major bodies of research relevant to our work here, multi-modal and multi-domain adaptation techniques, techniques for generating region proposals and object detection with RGB-D images.

Region Proposals We note that many top-performing supervised object detection methods [59] and weakly supervised methods [163, 85] rely on a good set of bottom-up bounding box object candidates. Object proposal generation has been an active area of research in computer vision in recent years [4, 102, 197, 178]. Given the importance of good region proposals [86], naturally people have studied the problem of using depth images to improve the quality of object proposals [119, 71]. Gupta *et al.*[71] use depth information to obtain improved contours from RGB-D images, and use this in a multi-scale combinatorial grouping framework [4] to report great improvements over RGB only methods, obtaining the same recall with an order of magnitude fewer regions as compared to RGB only methods.

RGB-D Detection Depth and RGB modalities often offer complementary information. Prior work has made use of this fact by producing detectors which take as input paired RGB and depth modalities to improve detection performance over the RGB only model [109, 165, 108]. Many of these methods do so by introducing new depth representations [90, 101, 172, 190, 183], most recently by adding an additional depth network representation into a convolutional network architecture [71, 70, 182].

[90, 101, 172, 190] propose extensions to deformable part based models [52] to compute additional features from the depth image, and report performance improvements over just using the RGB image. Song and Xiao [165] design rich features on the depth images while Gupta *et al.*[71] proposed a novel geocentric embedding for learning features from depth images, and both these methods report great improvements over previous works. While all of these methods report significant improvements over RGB-only methods, they all require bounding box annotations to train their models.

In Chapter 19, we will build off the ideas of LSDA (Part V) to allow us to adapt a CNN model trained for one task, which has plentiful training data, to perform a different test time task which has limited training data. Our work presented in Chapter 20 is inspired by RGB-D detection approaches, which successfully learn complementary depth feature representations. Therefore, we learn such representations at training time and learn to transfer information from the depth representation to an RGB only model through modality hallucination. We use depth side information at training time to transfer information through a new representation to our test time RGB model.

Transfer Learning. Our work is related to transfer learning and domain adaptation which learns to share information from one task to another. Classical approaches consider learning to adapt across distributions, through some combination of parameter updates [7, 40, 83] and

transformation learning [104, 63]. Christoudias *et al.* [27] learned a mapping to hallucinate a missing modality at training time, but was only shown with weak recognition models. Along these lines a transformation learning approach was recently introduced to use depth information at training time to inform RGB test time detection by learning transformations into a common feature representation across modalities [24]. In contrast to our approach, this paper learned a single representation for the joint modality space, while our work focuses on learning an additional RGB representation which is informed during training by the depth data. Such modality hallucination was explored in [166], which introduced a fusion approach which was able to fill in a missing modality.

Transferring Information Across Tasks Multi-modal deep learning architectures have been explored previously in a generative context [134, 167], and parallel convnet architectures have been previously explored in the context of Siamese network learning [23, 26]. Given the ease of collecting annotations for an image classification task, as opposed to an object detection task, there have been many techniques proposed to train detectors from weak labels [163, 3, 2, 183]. These method are notoriously hard to optimize and must be trained independently for each detection category. In Part V we proposed to transfer generic information from CNN based detectors to transform CNN classifiers into object detectors. Although effective, it was limited to transferring information between RGB models. Other approaches have been proposed to transfer generic information across modalities [27], but have only been shown with weak detection models.

Learning using side information. Our problem can also be viewed from the learning with side or privileged information perspective. This is when a learning algorithm has additional knowledge at training time, whether meta data or in our case an additional modality. One then uses this extra information to inform training of a stronger model than could be produced otherwise. The theoretical framework was explored in [179] and a max-margin framework for learning with side-information in the form of bounding boxes, image tags, and attributes was examined in [155], while Shrivastava and Gupta [156] showed how surface normals at training time could produce detection improvement within the DPM framework.

Network transfer through distillation. Most related to our work is the concept of network distillation and its extensions. Hinton *et al.* [74] and concurrently Ba *et al.* [8] introduced the idea of model compression and fast transfer of information from one convolutional network to another. Essentially, the output from one network is used as the target probability distribution for a new network. This was shown to reduce training time of a new network and in some cases reduce the number of parameters needed in order to achieve equivalent performance. This approach was further applied for transferring task correlation across domains, as discussed in Chapter III. Wang *et al.* [184] transfered information across networks without labels by used a ranking loss across video frames to learn a deep represen-

tation which mapped patches from the same track closer together than patches from distinct tracks.

Our approach can also be seen as using distillation to learn representations on RGB images by transferring supervision from paired depth images, but we employ joint training instead of staged training as was used in [69] for supervision transfer. In contrast to [69], our focus is different, we are studying the problem of enriching RGB representations using depth as side information. We show the result that learning representations using depth as side information in this manner can lead to representation which when used in conjunction with representations learned on ImageNet lead to boosts in performance for recognition tasks like object detection.

Chapter 19

Limited Depth Training Data

19.1 Introduction

Most well performing methods rely on Convolutional Neural Networks (CNNs) to learn features for depth images and require a large amount of annotated examples to be effective. Numerous efforts in the vision community over the last 15 years have led to the development of large scale RGB datasets [35, 49, 120], which have enabled huge progress on a variety of problems. However, while labeled RGB data is currently available for hundreds of categories with strong annotations and for thousands with weak annotations, the available labeled depth data is currently limited to tens of categories.

At the same time, the introduction of low cost and easy to use RGB-D image capturing systems has enabled many robotic setups to have access to both RGB and depth information during operation. Current techniques require bounding box annotations to train object detectors and limit use of depth images to categories for which such annotations exist. Thus, even though a depth sensor is available at test time, researchers are forced to use RGB-only detectors for most object categories they may want to study. This situation presents us with an interesting question: are detailed bounding box annotations for all object categories necessary to enable improved test time recognition using additional modalities. Or is there a way to utilize the vast amounts of labeled RGB data already available, along with limited labeled depth data, to train object detectors which can use RGB-D images at test time to boost performance over an RGB detector, even for objects with no labeled depth examples?

In this work, we address this question and propose a transfer approach which leverages labeled RGB-D data for some categories (denoted as auxiliary categories) to build RGB-D object detectors for additional categories for which we only have RGB training data (see Figure 19.1). We do this by fusing mid-level representations from depth and RGB images. This fused mid-level representation can be used with RGB-only classifiers to improve the quality of the RGB detector.

We evaluate our technique on the challenging NYUD2 dataset and our experiments show that we are able to effectively adapt RGB detectors into RGB-D detectors. These RGB-D

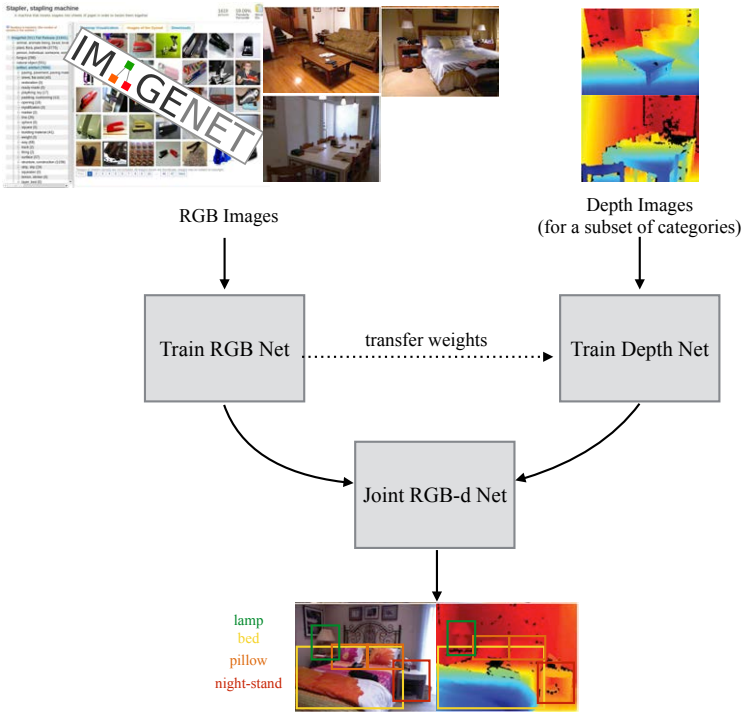


Figure 19.1: Given labeled depth images for a handful of categories we adapt an RGB object detector for a new category such that it can now use depth images in addition to RGB images at test time to produce more accurate detections. We do this by fusing information across modalities and use the available labeled depth data to extract mid-level depth representations which can be processed into semantic class labels for improved test time recognition performance on all categories of interest.

detectors can effectively leverage depth data at test time and we observe a 21% relative improvement over an RGB-only detector. Note that this was done without using any depth training data for the evaluated categories. We believe that our technique will facilitate the transfer of progress made in computer vision to fields like robotics.

19.2 Multimodal Architecture with Generic Depth Information

In this section, we describe our method for learning object detection models that use depth information from auxiliary categories to improve test-time performance for a new category.

We use $\mathcal{L}$ to denote the set of auxiliary categories for which we have annotated RGB-D data (bounding boxes around instances of the object in RGB-D scenes). We use $\mathcal{U}$ to denote the set of categories for which we only have labeled RGB data (again bounding boxes around

instances of the object in RGB scenes). Our goal is to leverage depth representations learnt by training RGB-D detectors for auxiliary categories $\mathcal{L}$ to adapt RGB object detectors for categories $\mathcal{U}$ to RGB-D input, that is they can now start using RGB-D input and potentially generate better output.

Intuitively, our method uses labeled depth training data for auxiliary categories $\mathcal{L}$ to learn a mid-level representation for depth images, which can be combined with mid-level representation from RGB images at test time. This mid-level fusion of representations can be used to adapt and improve a RGB object detector for the set of categories $\mathcal{U}$. The resulting RGB-D detector is able to utilize the depth data provided at test time to improve detection, without ever being trained on any depth data for categories $\mathcal{U}$.

Most state-of-the-art object detection models follow a two stage approach:

1. Computing region proposals: These are bounding boxes on the image which have high overlap with objects in the image.

2. Scoring region proposals: This is typically done by using CNNs [59, 72, 136, 170]. CNNs learn hierarchical feature representations in an end-to-end manner.

Our proposed technique incorporates depth information into both stages of this pipeline. For region proposals, we experimented with an adaptation of Edge Boxes [197] to depth images and RGB-D MCG [71]. We found RGB-D MCG to perform better and hence use these.

Next, we describe our technique for training multi-modal CNN based architectures with incomplete training data from one modality. In our case, we have complete RGB training data and limited depth training data.

Incorporating Depth into the CNN Representation

Our key insight is to fuse representations from RGB and depth images at an appropriate mid-level. Given a pair of RGB and depth images of a scene, the visual concepts depicted in both images are the same, though the pixel values may differ significantly. This motivates a processing pipeline which allows independent domain specific processing to arrive at a common mid-level representation, which can then be processed domain agnostically to obtain the desired semantic output. Thus, the domain specific learning can happen in the lower layers. These lower layers are often category agnostic (but domain specific) and can be trained effectively using data from a small set of categories, and can then be used with category specific but domain agnostic higher layers trained in a different domain or modality. Recent work on analyzing CNN architectures [113] in-fact shows quantitative evidence towards domain specific lower layers and task or category specific higher layers. To operationalize these findings, we use labeled RGB-D data from categories $\mathcal{L}$ to learn the domain specific but category independent lower layers and we use category specific but domain agnostic higher layers to obtain detectors for categories which lack labeled data in one of the modalities ($\mathcal{U}$).

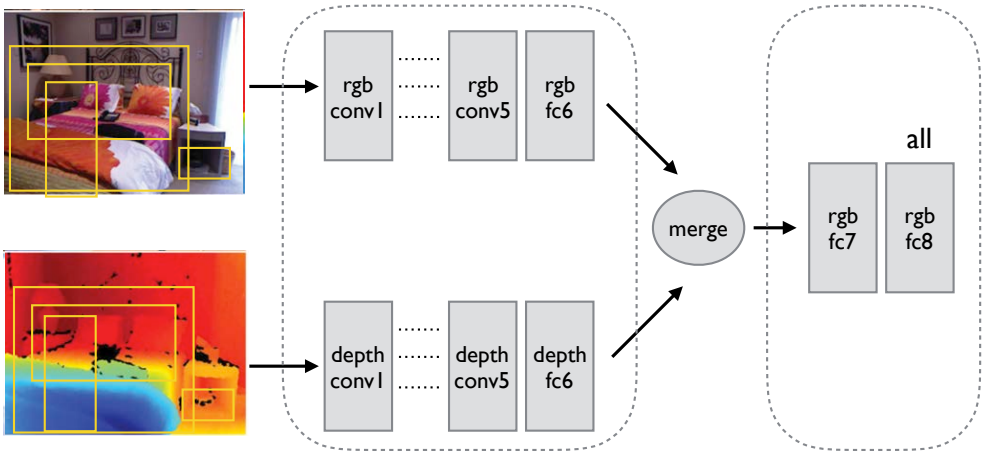


Figure 19.2: Our CNN architecture. We have parallel modality-specific lower layers and merge the two branches at a semantically meaningful higher layer.

Our proposed multi-modal architecture is depicted in Figure 19.2. We work with the popular AlexNet architecture [103]. AlexNet has five convolutional layers, three max pooling layers, and three fully connected layers. We use this architecture as a starting point for both the RGB and depth branches. Our insights about mid-level fusion and our training procedure are independent of the base CNN and should naturally extend to other CNN architectures.

It has been shown that the activations from layers fc6 and fc7 (the fully connected layers) produce semantically meaningful embeddings [40, 13]. We thus experimented with various fuse points in the fully connected layers, and found that fusing at fc6 worked better than both spatial fusion at pool5 and late fusion after fc7 (Section 19.3). For fusion we average the fc6 activations, after relu, of both branches and connect them with the 4096-dimensional fc7 layer, which is in turn connected to our final fc8 classifiers. We experimented with both average and concatenation as fusion techniques and found average to be slightly more robust.

Sequential Fine-Tuning

With the network structure determined, we now describe our method for training the network parameters. Since we lack depth training data for all categories in $\mathcal{U}$, we cannot naïvely fine-tune the full network. Instead, we propose a sequential fine-tuning procedure whereby the parameters of the RGB and depth networks are learned independently using all available labeled data from each modality.

Our training procedure is illustrated in Figure 19.1. We begin by training an RGB network (with AlexNet architecture), using labeled RGB data from all categories ($\mathcal{U} \cup \mathcal{L}$). We follow the standard practice of initializing this network from one that was pre-trained on the ImageNet dataset [35] for the task of image classification [40].

Next, we would like to produce an identical architecture that uses depth input in the form of an HHA encoding [71] (which encodes a depth image geocentrically using three channels: horizontal disparity, height above ground, and angle between the pixel's local surface normal and the inferred gravity direction). However, since we only have depth training data for categories in $\mathcal{L}$, we can not fine-tune the network from scratch.

Instead, we begin by populating all the weights of our depth network using the fully trained weights of our RGB network. By doing so, we initialize our depth network with parameters which have been tuned to perform well on all categories of interest, and in particular categories for which there is no depth training data. Additionally, initializing the depth network with RGB weights enables a favorable alignment between the two networks so they may be effectively combined later.

We next fine-tune the depth network on all available depth training data, allowing it to adapt to the new depth modality. Fine-tuning from RGB to depth HHA images is possible because the two modalities have similar structures [71] and higher level semantic information (e.g. object boundary information) is present in both.

Finally, after both the RGB and depth networks have been fine-tuned, we produce the final multi-modal network parameter values. For layers before the merge point, we transfer the weights from the RGB and depth networks directly to the corresponding weights of our architecture. For all layer weights above the merge point, we use the RGB model weights. This corresponds to reversing the upper depth weights back to their initialization point. We do this since the RGB parameters were learned using all labels for the portion of the model which processes mid-level representations into the final semantic outputs as opposed to the trained depth layers which have no recognition of the held out categories in $\mathcal{U}$.

19.3 Experiments

Dataset and Setup

We evaluate our algorithm with the NYUD2 dataset [158], using the standard split of 795 training images and 654 testing images. The split is selected such that images from the same scene do not co-occur in both sets. For all our experiments, we use annotations of the 19 major furniture categories: bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, garbage-bin, lamp, monitor, night-stand, pillow, sink, sofa, table, television, and toilet.

For all algorithms we use RGB-D MCG proposals [71]. MCG [4] generates a multi-scale hierarchical segmentation which is then used to generate region proposals. The proposals are then ranked by random forest regressors trained on features computed from the image and the region shape. Gupta *et al.*[71] generalized this to RGB-D images by using improved edge maps [71, 39, 68] and using features from the depth image in addition to features from the RGB image and the region shape for re-ranking the proposals. RGB-D MCG produces state-of-the-art region proposals for RGB-D images and we use these for our experiments.

method	modality	bath tub	bed	book shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tele vision	toilet	mean
Girshick *et al.*(Fast R-CNN) [60]	RGB	7.9	51.2	37.0	1.5	31.3	35.4	9.4	22.4	28.9	19.3	31.0	35.9	24.1	26.4	24.6	39.7	16.6	32.9	53.5	27.8
Our method merge fc6	RGB + aux D	9.7	64.1	37.4	2.1	40.2	44.8	11.9	21.7	39.2	27.8	35.4	46.8	40.2	36.8	27.9	48.4	22.8	35.5	49.0	33.8
Oracle merge fc6	RGB + D	4.7	73.3	45.6	3.6	45.2	54.6	16.8	26.1	47.1	34.9	40.8	49.7	51.7	41.6	39.3	55.7	27.3	48.5	64.4	40.6
Gupta *et al.* [70]	RGB + D	39.4	73.6	38.4	5.9	50.1	47.3	14.6	24.4	42.9	51.5	36.2	52.1	41.5	42.9	42.6	54.6	25.4	48.6	50.2	41.2
Gupta *et al.* [71] + Fast R-CNN	RGB + D	37.1	78.3	48.5	3.3	45.3	54.6	21.9	28.5	48.6	41.9	42.5	60.6	49.2	43.7	40.2	62.1	29.2	44.3	63.6	44.4

Table 19.1: **RGB-D Detection (mean AP%) on NYUD2 test set:** We compare our performance against several state-of-the-art methods. All methods use the AlexNet architecture. To report performance of our method 'RGB + aux D' for a particular category c, we use RGB data from all 19 categories and depth data from the remaining 18 'auxiliary' categories for training. We see our method is able to improve performance on held out categories from 27.8% to 33.8%.

In addition, all variants of our algorithm as well as all baseline and state-of-the-art results are reported using the AlexNet architecture, pre-trained with ImageNet RGB classification data. For our detection pipeline, we use the recently proposed Fast R-CNN [60] algorithm. We train both the RGB and depth networks each for 40,000 iterations with learning rate 0.001, momentum 0.9, and weight decay 0.0005 using the standard deep learning software package, Caffe [92].

RGB-D Detection

We begin by evaluating our algorithm on the NYUD2 test set for the RGB-D detection task [71]. Since we would like to understand the ability of our algorithm to produce an RGB-D detection model when no depth data is available for direct training, we perform hold one category out experiments. We perform 19 experiments where in the i^{th} experiment we remove labeled depth data corresponding to the i^{th} category when training [1] (so the detector has access to RGB data from all 19 categories and depth data from only 18 categories). We then use these detectors to report the AP obtained on the i^{th} category. The performance obtained by our method is reported in Table 19.1 under the name 'RGB + aux D'.

We compare against both the Fast R-CNN [60] RGB-only baseline as well as the state-of-the-art RGB-D detection models from Gupta *et al.*([70] and [71] + Fast R-CNN as described in [69]). Note that the later algorithms require full RGB and depth annotations and as such serve as an upper bound performance for our detection scenario. For reference, we also train our network using full RGB-D training data and report the performance as the oracle for our method (see Table 19.2). This number is expected to be slightly lower than competing state-of-the-art methods since our overall architecture ignores the semantic information learned in

[1] We do this by removing all bounding box proposals that overlap with the ground truth boxes for category i by any amount, though due to the small dataset size we continue to use regions from the image which do not overlap with the held out category. Note that a held out object may appear within the receptive field of another completely non-overlapping positive object or background box proposal due to the large size of pool5 receptive fields, but there is no supervision for the held-out category.

the highest layers of the depth network. This is necessary for the held out depth scenario, but is limiting in the full annotation scenario.

Overall, our method achieves 33.8% mAP when averaged across each independent held out category. In comparison RGB only model (but with the same MCG RGB-D proposals) only obtains a mAP of 27.8%. This shows that our mid-level fusion of RGB and depth is able to extract meaningful depth information which can be effectively combined with the RGB information to improve the eventual labeling function.

Ablation Study

In this section, we perform an ablation study on the architecture merge layer selection. For this experiment we further split the training set into the standard train/val sets, training with the train set and evaluating on the validation set. Table 19.2 reports results on the NYUD2 validation data set for our algorithm while varying the merge point of the RGB and depth networks. We select between the spatially aware pool5 layer and the higher, more semantically meaningful, fully connected layers, fc6, fc7, and fc8 (for oracle only).

We run our algorithm using the same experimental setup of holding out depth training data for one category at a time. For reference, we additionally report the performance of the oracle full depth trained network using each of these merge point selections. We find that merging the RGB and depth networks after fc6 provides the most benefit over using the RGB-only network. Since the depth network was trained only on the auxiliary 18 object categories, all category specific information which has been stored in the fc7 parameters serves as a distraction when attempting to detect the held out category.

In contrast, the oracle network performs best when merged after fc8, in other words a pure late-fusion approach. This is because the category specific parameters are relevant for all categories we wish to detect and are complementary to the RGB category specific parameters and aid the detection model at test time. Note that this is slightly different than the performance for Gupta *et al.*[71] + Fast R-CNN reported in Table 19.1. In our experiments the depth network was finetuned from the RGB network already finetuned on NYUD2 RGB images, as opposed to Gupta *et al.*[71] + Fast R-CNN which was finetuned from ImageNet classification weights.

Error Analysis

To investigate how our method uses depth to improve detection, we analyze the false positive errors made by our RGB-D detectors as compared to the baseline RGB-only and oracle fully supervised RGB-D detectors.

We know from Table 19.1 that our algorithm has fewer false positives overall than the RGB baseline and has more false positives than the oracle fully supervised RGB-D model. For further insight, we analyze the change in each type of false positives between our method and the baseline and between the oracle and baseline methods (see Figure 19.3). More precisely, for a given category, i, which has K ground truth instances in the test set, we look

method	modality	merge point	bath tub	bed	book shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tele vision	toilet	mean
Baseline [60]	RGB	-	9.2	44.1	11.6	1.4	24.4	25.0	6.8	17.8	15.0	18.6	18.1	42.1	25.3	16.3	19.6	21.0	13.6	35.5	58.5	22.3
Ours	RGB + aux D	pool5	8.4	50.2	3.4	2.1	26.8	25.7	3.6	10.0	23.9	33.5	14.1	38.7	36.1	23.3	20.1	28.8	14.8	31.6	61.6	24.0
Oracle	RGB + D	pool5	11.7	57.7	5.6	2.7	29.9	31.9	4.5	13.0	28.4	42.8	30.3	39.6	39.6	32.5	24.0	33.8	18.3	32.7	63.8	28.6
Ours	RGB + aux D	fc6	8.4	54.0	11.0	1.7	27.5	28.6	6.8	16.8	27.1	30.8	20.3	46.0	40.5	24.0	22.6	30.8	17.3	36.6	64.6	27.1
Oracle	RGB + D	fc6	14.3	66.1	16.9	3.0	36.4	39.3	6.8	20.2	31.9	39.2	31.6	45.1	48.1	32.8	28.6	38.9	22.9	37.7	69.1	33.1
Ours	RGB + aux D	fc7	4.7	54.3	6.3	1.1	26.4	26.4	5.7	9.3	27.6	21.9	15.2	44.2	35.6	15.6	8.8	28.8	16.3	35.8	54.0	23.1
Oracle	RGB + D	fc7	14.9	67.0	19.7	3.0	37.5	38.9	8.2	18.3	31.9	34.0	35.0	45.4	50.3	36.3	30.9	41.4	22.8	37.5	71.2	33.9
Oracle	RGB + D	fc8	15.4	70.6	21.6	3.7	37.4	38.2	8.8	17.4	31.1	34.4	36.7	43.6	50.7	37.5	30.2	40.4	22.9	38.1	71.5	34.2

Table 19.2: **RGB-D Detection (mAP%) on NYUD2 val set:** We compare various architecture merge points. All methods use the AlexNet architecture and hold depth training data out for the category being studied. We find that merging after fc6 performs the best on this dataset for the missing data setting. However, when all depth training data is available, late fusing at the scores is the best option.

at the top K scoring regions across the test set from the category i detector from the baseline RGB-only model, our model, and the oracle RGB-D model. For each model we compute the percent of the top K detections which correspond to each type of false positives. We then plot the difference in this percentage between the baseline and our method and the baseline and the oracle. For ease of viewing, categories are sorted per false positive type from least improvement of our method to most improvement by our method.

By studying these changes in the false positives, some interesting trends emerge. For instance, we find that our approach provides a relatively consistent improvement in localization and confusion with other categories (most bars in the *top row* are greater than or close to zero). In contrast, our method improves only 11/19 of the categories in confusion with background and hinders performance for the other 8/19 categories. We see that the oracle method provides improvement in confusion with background for almost all categories, which indicates there is potential to further improve these types of errors when RGB-D training data is available for the category of interest.

One interesting category is *television*, which has over a 15% reduction in the confusion with background when using our algorithm over the RGB baseline, but simultaneously has almost a 15% increase in the confusion with other category false positives. This is likely due to the fact that *monitor* is another category available and since during depth training the held out category *television* is not seen at the same time as the known category, *monitor*, this makes it harder for our algorithm to disambiguate the two categories at test time. This issue is mitigated with full supervision training of the depth net.

Finally, we show some qualitative examples of the improvements made by our approach. We pick the two categories where our method improves the most and least over the baseline. Figure 19.4 shows random images which contain bed and night-stand (categories where we improve the most 12.9% and 16.1%) where the top scoring detection is a true positive for our method and false positive for the baseline. Similarly in Figure 19.5 we show random images

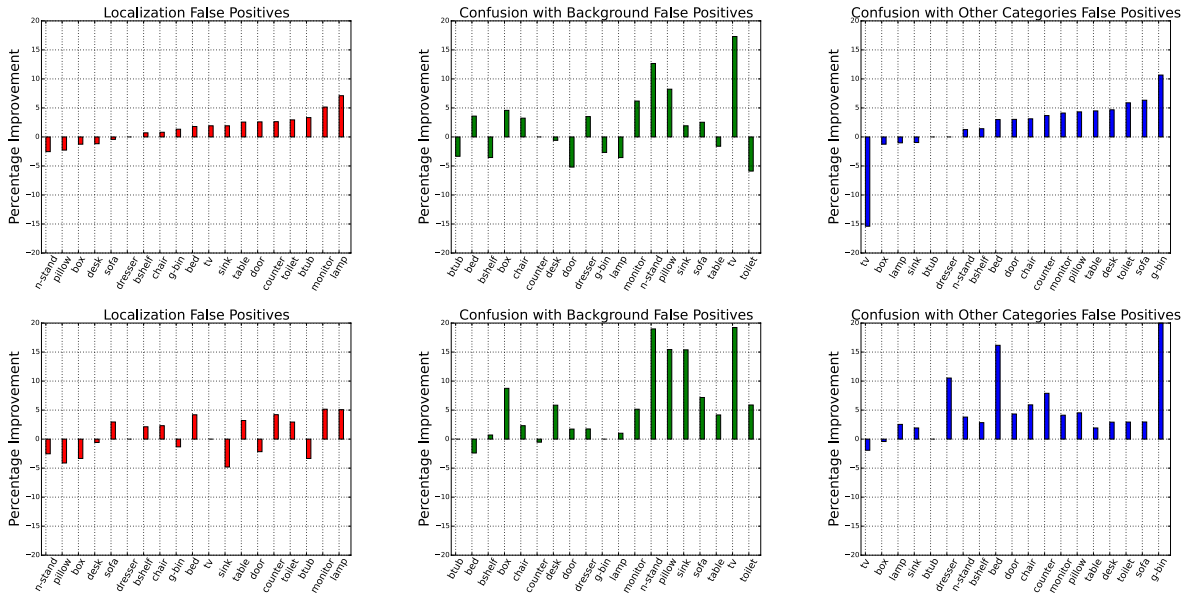


Figure 19.3: We study the change in the type of false positives between baseline and our method (*top row*) and the the change in the type of false positives between the baseline and the oracle for our method (*bottom row*). We show here false positives due to localization errors (red - left), confusion with background (green - center), and confusion with other categories (blue - right).

containing toilets and doors (categories where we improve the least -4.5% and -0.7%) where the highest scoring detection is a true positive for the baseline while it is false positive for our method.

In Figure 19.4, we very clearly see the effects of our method improving localization errors as well as fixing confusion with other categories. Similarly, Figure 19.5, provides examples where we begin to confuse with non-objects (background) for the toilet and door categories. With the exception of one of the toilet examples (middle) which is simply a result of the baseline region being just over threshold for overlap with ground truth to be considered a true positive, while our method's top scoring example was just under the threshold.

Large Scale RGB-D Detection

One of the main motivations behind our work is to enable enhanced RGB-D detection of a large number of objects with no depth training data, for applications such as robotics. We demonstrate the potential impact of our work by using our algorithm to extend the released 7.6k RGB detector [85] into an RGB-D detector, and show qualitative results in Figure 19.6. The LSDA [85] model was available only for RGB detection along with an RGB region proposal method (selective search [178]). We show results for the model from [85]

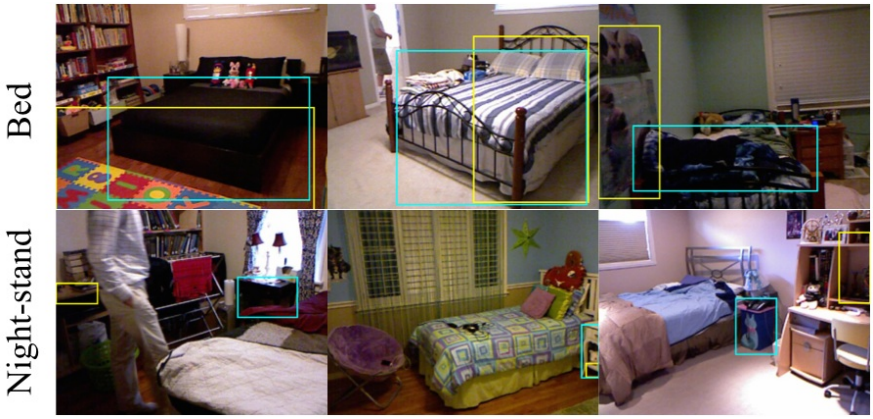


Figure 19.4: Example detections on the NYUD2 test set where the top detection from our method for the specified category is correct while the top detection from the RGB only baseline is incorrect. Cyan boxes are from our method and yellow boxes are from the RGB baseline.

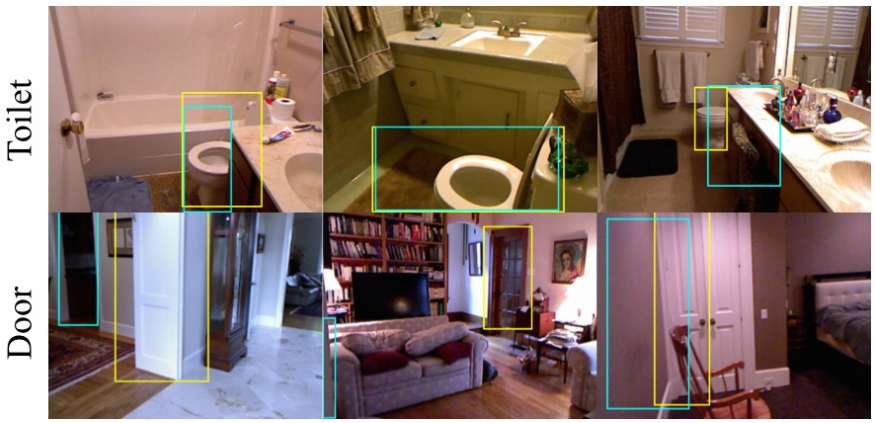


Figure 19.5: Example detections on the NYUD2 test set where the top detection from the RGB only baseline for the specified category is correct while the top detection from our method is incorrect. Cyan boxes are from our method and yellow boxes are from the RGB baseline.

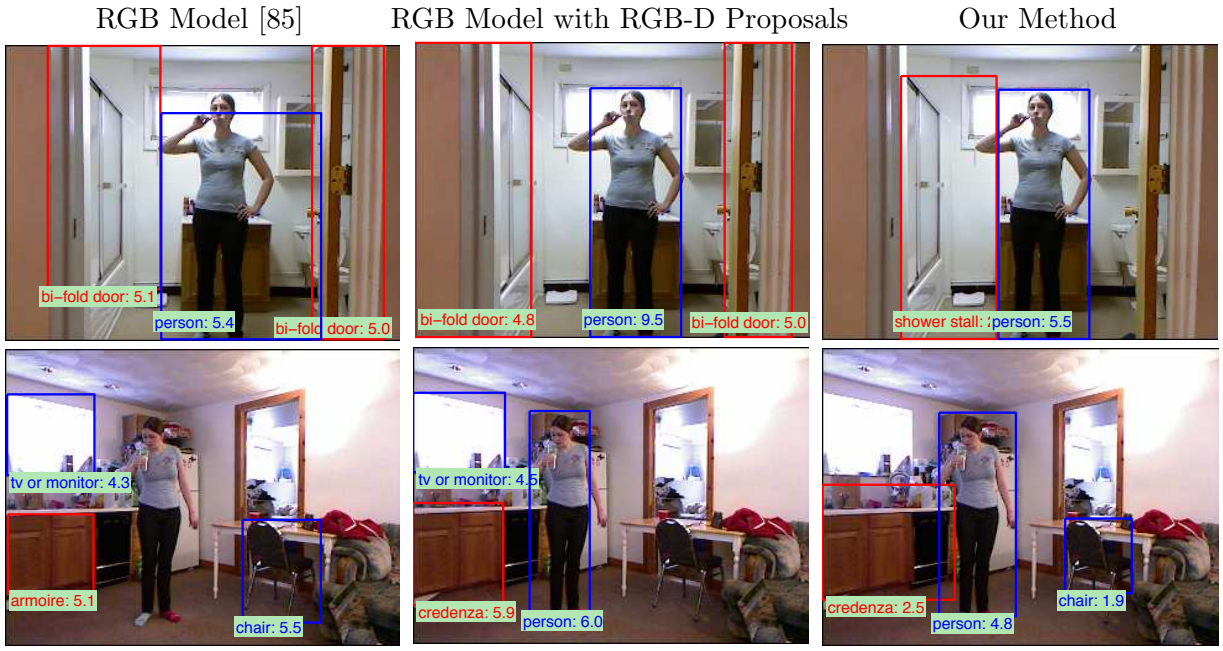


Figure 19.6: We use our algorithm to transform the publicly available 7.6k class RGB detector [85] into an RGB-D detector. We show here detection results for all 7.6k categories on example RGB-D images taken from two scenes in the Cornell activity dataset [169]. We present top detections from the original RGB CNN with RGB selective search region proposals (*left*), detections when using RGB-D MCG proposals (*middle*), and detections after our proposed adaptation (*right*). Blue boxes are detections of the 200 ILSVRC categories, while the red boxes are detections of the 7.4k categories corresponding to leaf nodes in the ImageNet database. Our algorithm not only provides better localization, but even enables extra categories to be detected.

in the *left* column. Next, we use the network parameters from the model from [85] along with RGB-D MCG proposals, as used throughout our method – the results are displayed in the *center* column. Finally, we produce a joint RGB-D network through our method of mid-level representation fusion and show results for our algorithm in the *right* column.[2] We show results on images taken from two scenes in the Cornell activity dataset [169], which contains categories not available during training on NYUD2 data, such as person.

After changing the region proposal mechanism to incorporate depth information, we see significant improvement in object localization. Upon using our algorithm to transform the RGB network into an RGB-D network, we see that false positives are reduced and new objects are recognized.

[2]Note that these results were obtained using the publicly released LSDA R-CNN detector [85] and not the Fast R-CNN detector that is used for the rest of the experiments. We expect similar results with the Fast R-CNN based detector.

This qualitative result is highly encouraging as it demonstrates that our algorithm incorporates category invariant depth information that is generic enough to be useful with a detector that was trained on separate tasks and in a different data source. For example, people, shower stalls, and credenzas never appear in NYUD2 training annotations, where we train our depth model. However, we are able to learn to effectively combine the generic depth and RGB processing of the lower layers and use the modified intermediate representation as additional information for the category specific classification layer. This model was able to be produced without further RGB training, meaning that our pre-trained RGB detector could immediately be adapted to utilize depth information at test time. In the future we plan to conduct a more quantitative study of this results.

Chapter 20

Limited Depth Test Data

20.1 Introduction

While RGB image capturing devices are pervasive, depth capturing devices are much less prevalent for consumer applications. This means that many recognition models will need to perform well on RGB images alone as input. We present an algorithm which uses available paired RGB-d training data to learn to hallucinate mid-level convolutional features from an RGB image. We demonstrate that through our approach we produce a novel convolutional network model which operates over only the single RGB modality input, but outperforms the standard network which only trains on RGB images. Thus, our method transfers information commonly extracted from depth training data to a network which can extract that information from the RGB counterpart.

Convolutional networks (ConvNets) have produced tremendous success on visual recognition tasks, from classification [103, 159, 170], to detection [59, 152], to semantic segmentation [123, 196]. The standard approach for training these networks is to initialize the network parameters using a large labeled image corpora (ex: ImageNet [35]) and then fine-tune using the smaller target labeled data sources. While this strategy has been proven to be very effective, it offers only one technique for learning representations for recognition and due to the large parameter space of the network, runs the risk of overfitting to the nuances of the small RGB dataset.

We propose an additional representation learning algorithm which incorporates side information in the form of an additional image modality at training time to produce a more informed test time single modality model. We accomplish this by directly learning a modality hallucination network which optimizes over the standard class and bounding box localization losses while being guided by an additional hallucination loss which regresses the hallucination features to the auxiliary modality features.

Due to its practicality, we consider the case of producing an RGB detector using some paired RGB-D data at training time. In doing so, we produce a final model which at test time only sees an RGB image, but is able to extract both the image features learned through

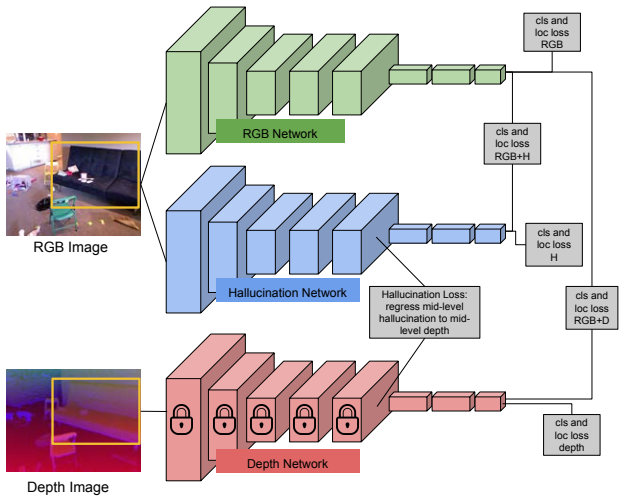


Figure 20.1: Training our modality hallucination architecture. We learn a multimodal Fast R-CNN [60] convolutional network for object detection. Our hallucination branch is trained to take an RGB input image and mimic the depth mid-level activations. The whole architecture is jointly trained with the bounding box labels and the standard softmax cross-entropy loss.

finetuning with standard supervised losses as well as the hallucinated features which have been trained to mirror those features you would extract if a depth image were present. We demonstrate that our RGB with hallucination detector model outperforms the state-of-the-art RGB model on the NYUD2 dataset.

20.2 Modality Hallucination Model

We present a modality hallucination architecture for training an RGB object detection model which incorporates depth side information at training time. Our hallucination network learns a new and complementary RGB image representation which is trained to mimic depth mid-level features. This new representation is combined with the RGB image representation learned through standard fine-tuning.

Figure 20.1 illustrates the training architecture for our hallucination model. We use multi-layer convolutional networks (ConvNets) as our base recognition architecture which have been shown to be very effective for many different recognition tasks. Prior work on RGB-D detection [71] has found success using a two channel model where RGB and depth images are processed independently with a final detection score being the softmax of the average of both predictions.

For our architecture we build off of this same general model. However, we seek to share information between the two modalities and in particular to use the training time privileged depth modality to inform our final RGB only detector. To accomplish this, we introduce a

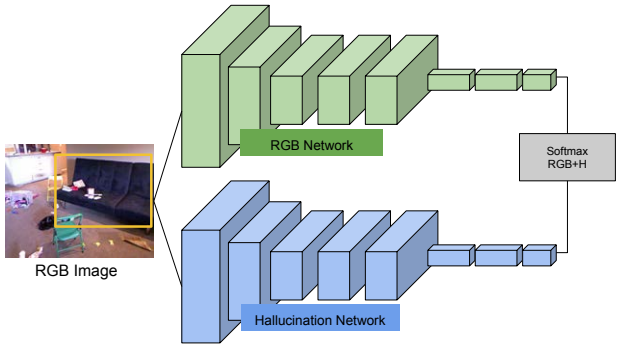


Figure 20.2: Test time modality hallucination architecture.

third channel which we call the *hallucination network* (blue network in Figure 20.1). The hallucination network takes as input an RGB image and a set of regions of interest and produces detection scores for each category and for each region.

To cause the depth modality to share information with the RGB modality through this hallucination network, we add a regression loss between paired hallucination and depth layers. This choice is inspired by prior work which uses similar techniques for model distillation [74], task correlation transfer across domains [177], and supervision transfer from a well labeled modality to one with limited labels [69]. Essentially, this loss guides the hallucination network to extract features from an RGB image which mimic the responses extracted from the corresponding depth image. We will discuss the details of this loss and its optimization in the next section. It is important that the hallucination network has parameters independent of both the RGB and depth networks as we want the hallucination network activations to match the corresponding depth mid-level activations, however we do not want the feature extraction to be identical to the depth network as the inputs are RGB images for the hallucination network and depth images for the depth network.

At test time, given only an RGB image and regions of interest, we pass our image through both the RGB network and the hallucination network to produce two scores per category, per region, which we average and take the softmax to produce our final predictions (see Figure 20.2).

20.3 Architecture Optimization

In this section we describe the implementation and optimization details for our architecture. At training time we assume access to paired RGB and depth images and regions of interest within the image. We train our model one set of paired images at a time using the Fast R-CNN [60] framework. The RGB and depth network are independently trained using the Fast R-CNN algorithm with the corresponding image input. Next, the hallucination network

parameters are initialized with the learned depth network weights before joint training of the three channel network. The choice of initialization for the hallucination parameters is explored in Section 20.4. Note, that finetuning of the hallucination network with only a softmax loss on the label space would be equivalent to the training procedure of the RGB network. To facilitate transfer we must use an additional objective by introducing a hallucination loss.

Hallucination Loss. We add the objective that activations after some layer, ℓ, should be similar between the hallucination and depth networks. In particular, we add a euclidean loss between the depth activations A_ℓ^{dNet} and the hallucination activations A_ℓ^{hNet} so that the hallucination loss for the given layer is defined as:

$$\mathcal{L}_{\text{hallucinate}}(\ell) \quad = \quad \|\sigma(A_\ell^{\text{dNet}}) - \sigma(A_\ell^{\text{hNet}})\|_2^2 \tag{20.1}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function.

This loss can be applied after any layer in the network and can be optimized directly. However, we are trying to learn an asymmetric transfer of information, namely we seek to inform our RGB hallucination model using the pre-learned depth feature extraction network. Therefore, we set the learning rates of all layers lower than the hallucination loss in the depth network to zero. This effectively freezes the depth extractor up to and including layer ℓ so that the target depth activations are not modified through backpropagation of the hallucination loss.

Multi-task Optimization The full training of our model requires balancing multiple losses. More precisely we have 11 total losses, 5 softmax cross-entropy losses using bounding box labels as targets, 5 Smooth L1 losses [60] using the bounding box coordinates as the targets, and one additional hallucination loss which matches midlevel activations from the hallucination branch to those from the depth branch. The 5 standard supervision and 5 bounding box regression losses operate over each of the three subnetworks, RGB, depth, hallucination, independently so that each learns weights that are useful for then final task. We then have 2 joint losses over the average of the final layer activations from both the RGB-depth branches and from the RGB-hallucination branches. These losses encourage the paired networks to learn complementary scoring functions.

For a given network, N, let us denote the softmax cross-entropy loss over category labels as $\mathcal{L}_{\text{cls}}^{\text{N}}$ and the Smooth L1 loss over bounding box coordinate regression as $\mathcal{L}_{\text{loc}}^{\text{N}}$. Then, the total joint loss of our optimization can be described as follows:

$$\begin{aligned} \mathcal{L} \quad = \quad & \gamma \mathcal{L}_{\text{hallucinate}} \\ & + \alpha \left[\mathcal{L}_{\text{loc}}^{\text{dNet}} + \mathcal{L}_{\text{loc}}^{\text{rNet}} + \mathcal{L}_{\text{loc}}^{\text{hNet}} + \mathcal{L}_{\text{loc}}^{\text{rdNet}} + \mathcal{L}_{\text{loc}}^{\text{rhNet}} \right] \\ & + \beta \left[\mathcal{L}_{\text{cls}}^{\text{dNet}} + \mathcal{L}_{\text{cls}}^{\text{rNet}} + \mathcal{L}_{\text{cls}}^{\text{hNet}} + \mathcal{L}_{\text{cls}}^{\text{rdNet}} + \mathcal{L}_{\text{cls}}^{\text{rhNet}} \right] \end{aligned} \tag{20.2}$$

method	btub	bed	bshelf	box	chair	counter	desk	door	dresser	gbin	lamp	monitor	nstand	pillow	sink	sofa	table	tv	toilet	mAP
RGB only [60] (A)	7.5	50.6	**36**.8	1.4	30.2	34.9	10.8	21.5	27.8	16.9	26.0	32.6	20.6	**25**.1	31.6	**36**.7	14.8	25.1	54.6	26.6
RGB ensemble (A-A)	10.5	53.7	33.6	1.6	32.0	34.8	12.2	20.8	34.5	19.6	28.6	45.7	28.5	24.4	31.4	34.7	14.5	**34**.0	**56**.1	29.0
Our Net (A-RGB, A-H)	**13**.9	**56**.1	34.4	**1**.9	**32**.9	**40**.5	**12**.9	**22**.6	**37**.4	**22**.0	**28**.9	**46**.2	**31**.9	22.9	**34**.2	34.2	**19**.4	33.2	53.6	**30**.5
RGB only [60] (V)	15.6	59.4	38.2	1.9	33.8	36.3	12.1	24.5	31.6	18.6	25.5	46.5	30.1	20.6	30.3	40.5	19.5	**37**.8	45.7	29.9
RGB ensemble (A-V)	14.8	60.4	**43**.1	**2**.1	36.4	40.7	13.3	**27**.1	35.5	20.8	29.9	**52**.9	**33**.5	26.2	33.0	44.4	19.9	36.7	50.2	32.7
Our Net (A-RGB, V-H)	**16**.8	**62**.3	41.8	**2**.1	**37**.3	**43**.4	**15**.4	24.4	**39**.1	**22**.4	**30**.3	46.6	30.9	**27**.0	**42**.9	**46**.2	**22**.2	34.1	**60**.4	**34**.0

Table 20.1: **Detection (AP%) on NYUD2 test set:** We compare our performance (pool5 hallucinate) against a Fast R-CNN [60] RGB detector trained on NYUD2 and against an ensemble of Fast R-CNN RGB detectors. AlexNet [103] architecture is denoted as 'A' and VGG-1024 [60, 159] architecture is denoted as 'V'. Our method outperforms both the RGB-only baselines and the RGB ensemble baselines.

Balancing these objective is an important part of our joint optimization. For simplicity, we choose to weight all localization losses equivalently and all category losses equivalently. This leaves us with three parameters to set, denoted above as α, β, and γ.

We set the category loss weights, $\beta = 1.0$, and then let the localization weights be a factor of 2 smaller, $\alpha = 0.5$. Finally, to set the hallucination loss weight will depend on the approximate scale of the loss function. This will vary based on the layer at which the hallucination loss is added. For lower layers in the network, the loss tends to be larger. Thus, a smaller value for γ would make sense to avoid the hallucination loss dominating the other objectives. We therefore use a heuristic that the contribution of the hallucination loss should be around 10 times the size of the contribution from any of the other losses. For example, if the contribution from a category loss is about 0.5, then the contribution from the hallucination loss should be around 5. In practice, one can determine this by running a few iterations of training and examining the losses.

Gradient Clipping In developing our model, we found that the optimization could be susceptible to outliers causing large variations in gradient magnitudes for the hallucination loss. One potential way to address this issue would be to set the loss weight very low on the hallucination loss so that even when a large gradient appears the network optimization does not diverge. However, this will limit the effectiveness of the hallucination loss.

Instead, we have found that a more robust way to train with this euclidean loss is to use gradient clipping. This simply means that when the total gradient (in terms of $\ell 2$ norm) in the network exceeds some threshold, T, all gradients are scaled by T / (total norm). Thus, the effective contribution of an outlier example is reduced since the large gradients will be scaled down to the standard range. This approach is simple and already implemented in many standard deep learning packages (ex: it involves a single line change in the Caffe [92] solver file).

20.4 Experiments

We evaluate our model using a standard RGB-D detection dataset, NYUD2 [190]. The NYUD2 dataset consists of 1449 labeled RGB-D images. The dataset is split into train (381 images), val (414 images), and test (654 images) sets. For our ablation experiments we train our model using the train set only and evaluate our model on the validation set. For our overall detection experiment which compares to prior work, we present results on the test set for our algorithm trained using the combined trainval set.

Base Network. For the following experiments our base network architecture (used for each of the RGB, depth and hallucination networks), is the single scale Fast R-CNN modification to the AlexNet [103] architecture or the VGG-1024 architecture introduced in [60] as a lower memory modification of VGG [159]. The RGB AlexNet network is initialized with the CaffeNet [92] released weights, which were learned using ILSVRC12 [35] and the RGB VGG-1024 network was initialized with the weights released with Fast R-CNN [60]. We then finetune our RGB network on the NYUD2 dataset. We represent the depth images using the HHA encoding introduced by Gupta *et al.* [71] and independently finetune the depth network after initializing with the RGB weights.

Region Proposals. A Fast R-CNN architecture takes as input an image and its corresponding regions of interest. To compute these regions of interest we use two different region proposal algorithms. For the NYUD2 dataset we use multi-scale combinatorial grouping (MCG) [4], which has been used in the past for this dataset as it is capable of incorporating depth information into the proposal mechanism. We use the RGB-D version of MCG for training all networks and then use the RGB version at test time. We found this to work better than using RGB MCG for both training and testing by about 1-2%.

SGD Hyper-parameters. We optimize our network using the Caffe [92] learning framework. We use a base learning rate of 0.001 and allow all layers of the three channel network to update with the same learning rate, with the exception of the depth network layers below the hallucination loss, which are frozen. We use a momentum of 0.9 and a weight decay of 0.0005. We optimize our ablation experiments for 40K iterations and our full NYUD2 experiment for 60K iterations[1] using a step learning rate policy where the base learning rate is lowered by a factor of 10 ($\gamma = 0.1$) every 30K iterations. Finally, we clip gradients when the L2 norm of the network gradients exceeds 10.

NYUD2 Detection Evaluation

Table 20.1 reports performance of our full system with two different architecture on the NYUD2 dataset. The two base architectures are either AlexNet (indicated as 'A') [103] or VGG-1024 (indicated as 'V') [60, 159]. We train our initial RGB and depth networks

[1]Note that for one of the initial RGB AlexNet models we use the weights released with [69] which was only trained for 40K iterations. We also note that in our experience training the RGB only AlexNet baseline model for more than 40K iterations did not provide any benefit as it does for the joint hallucination model and for the VGG-1024 architecture.

using the strategy proposed in [71], but use Fast R-CNN instead of RCNN as used in [71]. We then initialize our hallucination network using the depth parameter values. Finally, we jointly optimize the three channel network structure with a hallucination loss on the pool5 activations. When our hallucination network is labeled with a particular architecture this refers to the choice of the depth network and the hallucination network architecture and the RGB architecture is chosen and indicated separately. In the next two sections we explore our choice of initialization and at which layer to add a hallucination loss.

Initial Weights	bathtub	bed	bshelf	box	chair	counter	desk	door	dresser	gbin	lamp	monitor	nstand	pillow	sink	sofa	table	tv	toilet	mAP
RGB	7.5	50.4	9.9	**0**.9	**2**6.2	**2**4.9	5.8	**1**5.8	13.0	**2**9.8	12.0	43.1	20.9	14.7	17.9	25.3	**1**5.1	32.5	59.1	22.4
depth	9.9	**5**2.4	**1**4.9	**0**.9	24.9	24.4	4.3	15.3	18.1	24.1	**1**4.8	**4**5.8	**2**7.2	**1**8.5	**2**1.3	**2**9.0	13.7	**3**3.6	**6**6.4	**2**4.2
random	**1**0.5	47.6	12.3	0.6	23.5	20.2	**6**.0	13.0	**1**9.3	12.0	13.3	42.8	12.8	12.1	13.6	23.0	13.9	28.6	61.5	20.3

Table 20.2: **RGB Detection (AP%) on NYUD2 val set:** We compare intializing the hallucination network by randomly initializing or by using the pre-trained RGB or depth parameter values.

For each architecture choice we first compare against the corresponding RGB only Fast R-CNN model and find that our hallucination network outperforms this baseline, with 30.5 mAP vs 26.6 mAP for the AlexNet architecture and 34.0 mAP vs 29.9 mAP for the VGG-1024 architecture. Note that for our joint AlexNet method, A-RGB + A-H, we average the APs of the joint model using each of the AlexNet RGB baseline models. As an additional reference, the state-of-the-art performance of RGB-D detection algorithms on NYUD2 is 41.2 mAP [70], 44.4 mAP [71] when run with Fast R-CNN [60] and 47.1 mAP [69]. However, these algorithms operate in the privileged regime with access to depth at test time, thus they are able to achieve the highest overall performance.

It is well known that ensemble methods tend to outperform the single model approach. For example, an ensemble of two ConvNets each initialized randomly and then trained using the same data source, outperforms either model independently [67]. Since our method is the combination of an RGB model trained using a standard supervised approach and an RGB model trained using our depth hallucination technique, we additionally compare our approach to an ensemble of standard trained RGB models. Table 20.1 reports the performance both for an ensemble of two different AlexNet RGB models, the weights for which were randomly initialized with different seeds before being pre-trained with ImageNet [35], and for an ensemble of an AlexNet RGB model with a VGG-1024 RGB model. We find in both cases that the RGB ensemble improves performance over the single RGB model, while our hallucination model offers the highest performance overall, with 14/19 categories improving for the AlexNet comparisons to ensemble and 13/19 categories improving for the VGG-1024 hallucination net comparisons to ensemble. This suggests that our hallucination model offers more benefit than a simple RGB ensemble.

While our method hallucinates mid-level depth features, other work has proposed hallucinating the pixel level depth from an RGB image. As an additional baseline, we have taken a state-of-the-art depth estimation approach [121] and used the model to produce hallucinated depth images at test time which can be used as input to the depth channel of our pre-trained

RGB-D detector. However, doing this performed worse than using our RGB model alone (22% mAP vs 27% mAP) so we have omitted the results from Table 20.1. Note that we do not fine-tune our detection model using the depth pixel hallucinations and thus a drop in performance is likely due, at least in part, to the mismatch between the true depth used at training time and the hallucinated depth images used at test time. We refer the interested reader to a related and more comprehensive investigation of pixel depth hallucination by Eigen and Fergus [48] who replaced the true depth input into their network with their hallucinated depths and normals and did fine-tune, yet still did not observe performance improvements for the final semantic segmentation task.

In the next subsections we explore ablation studies and analysis on our hallucination model. For all the following experiments we use the AlexNet RGB and hallucination architecture.

How to initialize the hallucination net?

One important parameter of training our model is how to initialize the hallucination network. We explore three natural choices in Table 20.2, random initialization, initialization with the RGB network parameter values, and initialization with the depth network parameter values. Here we use RGB and depth networks trained using NYUD2 *train set* only and then we use the NYUD2 *validation set* for evaluation of the different choices. We find that both the RGB and depth initialization schemes outperform the baseline RGB only model (20.6% mAP for this setting) and the random initialization model. The depth initialization model has the highest mAP performance and higher AP than the RGB initialization model on 12/19 categories (plus 1 tied category). We thus choose to initialize our hallucination network in all future experiments with the depth parameter values.

hallucination layer	bathtub	bed	bshelf	box	chair	counter	desk	door	dresser	gbin	lamp	monitor	nstand	pillow	sink	sofa	table	tv	toilet	mAP
RGB only	4.9	45.5	10.9	1.3	21.5	23.6	5.4	14.5	12.7	17.4	9.4	40.9	17.2	14.9	19.9	19.2	14.0	32.5	66.3	20.6
pool1	**12**.0	**54**.0	**17**.9	**1**.1	24.5	23.6	5.0	15.2	16.3	12.7	13.3	40.0	24.7	16.6	20.5	**29**.6	14.9	27.4	55.3	22.3
pool2	8.4	50.7	13.5	1.0	24.2	26.0	6.6	13.1	13.8	17.8	11.7	40.7	21.8	15.0	20.5	22.4	15.2	27.2	59.7	21.5
conv3	8.8	52.5	13.2	1.0	25.6	26.2	3.3	13.2	14.9	17.0	**16**.2	41.6	22.2	**20**.2	22.9	24.6	**17**.2	**37**.4	65.6	23.3
conv4	9.7	51.2	12.9	1.0	26.3	**26**.8	**6**.9	**17**.4	16.7	22.0	12.4	43.2	15.5	16.4	**24**.0	23.5	16.2	34.2	64.2	23.2
pool5	9.9	52.4	14.9	0.9	24.9	24.4	4.3	15.3	**18**.1	**24**.1	14.8	**45**.8	**27**.2	18.5	21.3	29.0	13.7	33.6	**66**.4	**24**.2
fc6	10.3	47.2	12.0	0.6	21.7	20.0	5.9	12.8	13.8	20.5	11.8	34.4	16.3	13.1	14.8	27.3	16.1	28.8	60.5	20.4
fc7	3.3	49.4	12.7	0.8	24.1	21.8	4.8	15.2	16.8	11.7	10.0	43.4	18.7	14.2	20.6	25.2	14.4	29.5	63.1	21.0
fc8	4.2	50.7	13.9	0.9	23.8	23.6	5.4	15.5	18.0	13.2	13.3	42.0	20.9	15.8	22.3	23.8	14.5	29.6	63.6	21.8

Table 20.3: **RGB Detection (AP%) on NYUD2 val set:** We compare hallucinating different mid-level features with our method.

Which layer to hallucinate?

Another important parameter of our method is to choose which mid-level activations the hallucination loss should regress to. In Table 20.3 we systematically explore placing the

method	chair	dining table	sofa	tv	mAP
RGB	17.5	13.0	10.4	26.7	16.9
RGB+H	**19**.5	**17**.4	**19**.3	**27**.1	**20**.8
RGB (pascal ft)	33.1	**63**.5	49.1	62.7	52.1
RGB (pascal ft) + H (no ft)	**34**.3	61.9	**53**.3	**63**.9	**53**.4

Table 20.4: **RGB Detection (AP%) on PASCAL voc 2007 test set:** We compare running our hallucination network on a new dataset. We compare the RGB only vs hallucination network of NYUD2 by first directly applying the networks on pascal. Then we finetune the RGB model on pascal data (leaving the hallucination portion fixed) and continue to find that the nyud trained hallucination model provides performance improvements.

hallucination loss after each layer from pool1 to fc8. We found that overall adding the hallucination loss at a mid to lower layer improved performance the most over the RGB only baseline network.

The highest overall performance was achieved with the hallucination loss on the pool5 activations. However, the result was not uniformly distributed across all categories. For example, *bathtub* received a noticeably greater performance increase with a hallucination loss at pool1.

We also experimented with adding the hallucination loss at multiple layers in the network, but did not find this to be more effective than pool5 alone.

Does hallucination help on other datasets?

We next study the application of our hallucination network on the Pascal [49] dataset (VOC 2007) which lacks depth data. First, we directly evaluate both the NYUD2 RGB-only network and our NYUD2 RGB plus hallucination network on the four overlapping categories in Pascal. Results for this experiment are reported in the first two rows of Table 20.4.

We find that our hallucination network provides 3.9% mAP improvements across these four Pascal categories when compared to the RGB-only baseline (from 16.9 to 20.8 mAP). Additionally, we note that there is a dataset shift between Pascal and NYUD2 which causes the overall performance of both methods to be lower than that of a network which was explicitly trained on Pascal. Therefore, we also explore further fine-tuning on the available Pascal VOC 2007 trainval set. This set only contains RGB images so we may only further fine-tune the RGB network.

This means that the dataset shift is mitigated in the RGB network but not in the hallucination network. Nevertheless, we find that the combination of our Pascal fine-tuned RGB network with our NYUD2 trained hallucination network continues to outperform the RGB-only baseline, achieving 53.2 mAP instead of 52.1 mAP and higher performance on 3/4 categories.

This indicates that the hallucination technique provides benefit beyond the NYUD2 dataset and we expect that the gains from the hallucination network would only become larger if we were able to adapt the parameters to the new dataset directly.

What did the hallucination net learn?

Regression losses can often be difficult to train together with the supervised cross-entropy loss. We first verify that our hallucination loss is effectively learning by examining the training loss vs iteration and confirming that the hallucination loss does indeed decrease.

We next verify that this training loss decrease translates to a decreased loss on the test data and hence a better depth activation alignment. To this end, we examine the network outputs on the NYUD2 test set. We first compute the hallucination loss value across the entire test set before and after learning and find that the value decreases from 216.8 to 94.6.

We additionally compare the euclidean distance between the hallucination activations and the RGB activations and find that after learning, the hallucination and depth activations are closer than the hallucination and RGB activations. Specifically, for the case where the hallucination network was initialized with RGB weights, the hallucination network activations start out being same as the RGB network activations but over time become closer to the depth network as can be seen from the post-training euclidean losses of H-RGB =113.0 while H-HHA=97.5.m

As an example, Figure 20.3 shows roi-pool5 activations from corresponding regions in the test image which have highest final detection scores. The visualization shows all $256 \times 6 \times 6$ roi-pool5 activations and corresponding region label. This figure illustrates the difference between the RGB activations learned through our approach and through the standard learning procedure.

Finally, we know from the detection experiments in the previous section that training with the hallucination loss offers performance improvements over a single RGB model or an ensemble of RGB models trained without the depth hallucination loss. However, it's important to know how the network is improving.

Therefore, in Figure 20.4, we show randomly sampled images from the NYUD2 test set where the top scored region from our hallucination model corresponds to a true positive and the top scoring region from the single RGB baseline corresponds to a false positive. Our method output is illustrated with a green box and the baseline is illustrated with a red box.

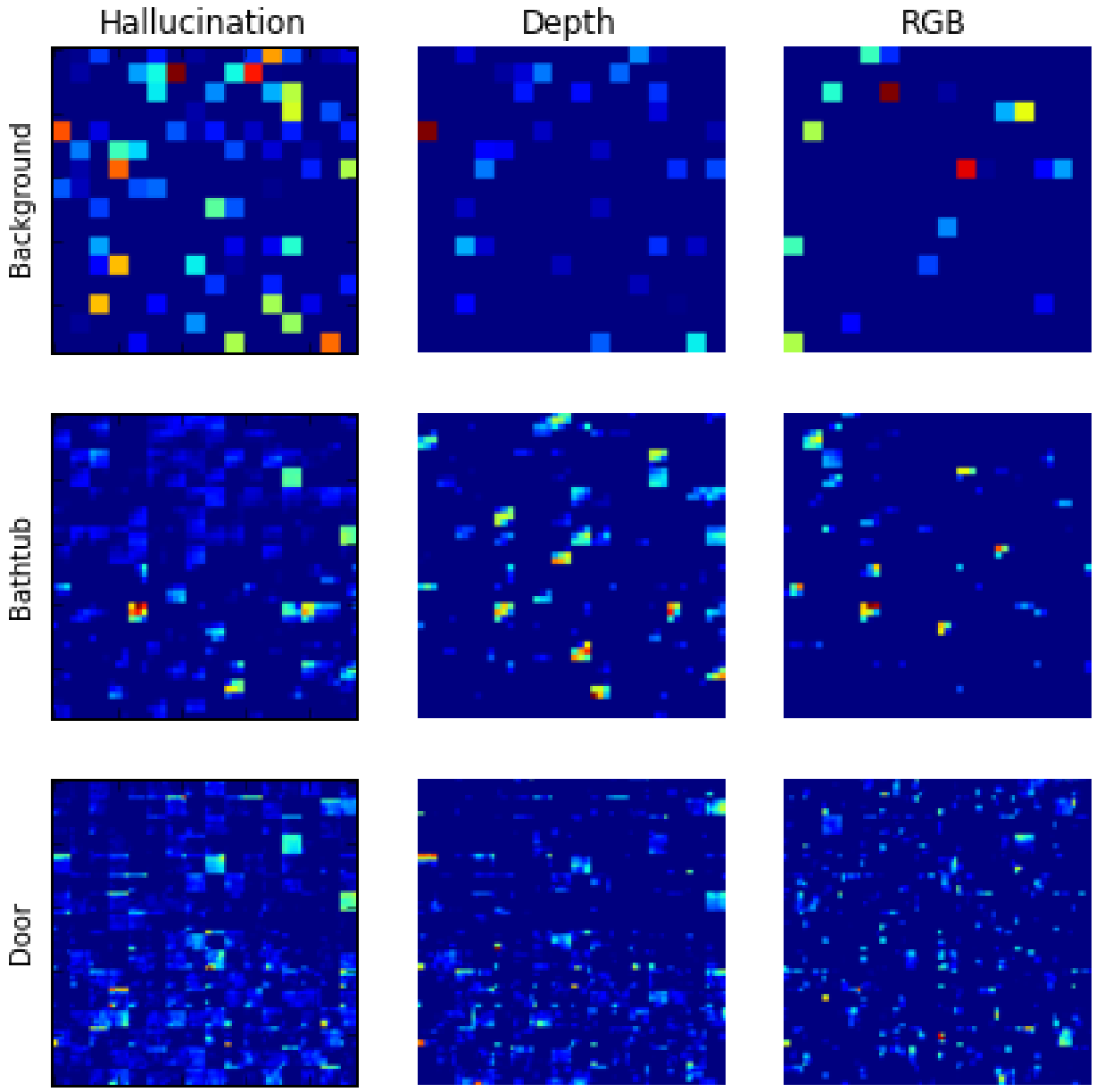


Figure 20.3: **Roi-pool5 activations** on three top scoring regions from an NYUD2 *test set* image. This figure illustrates the difference between the activations from the three networks.

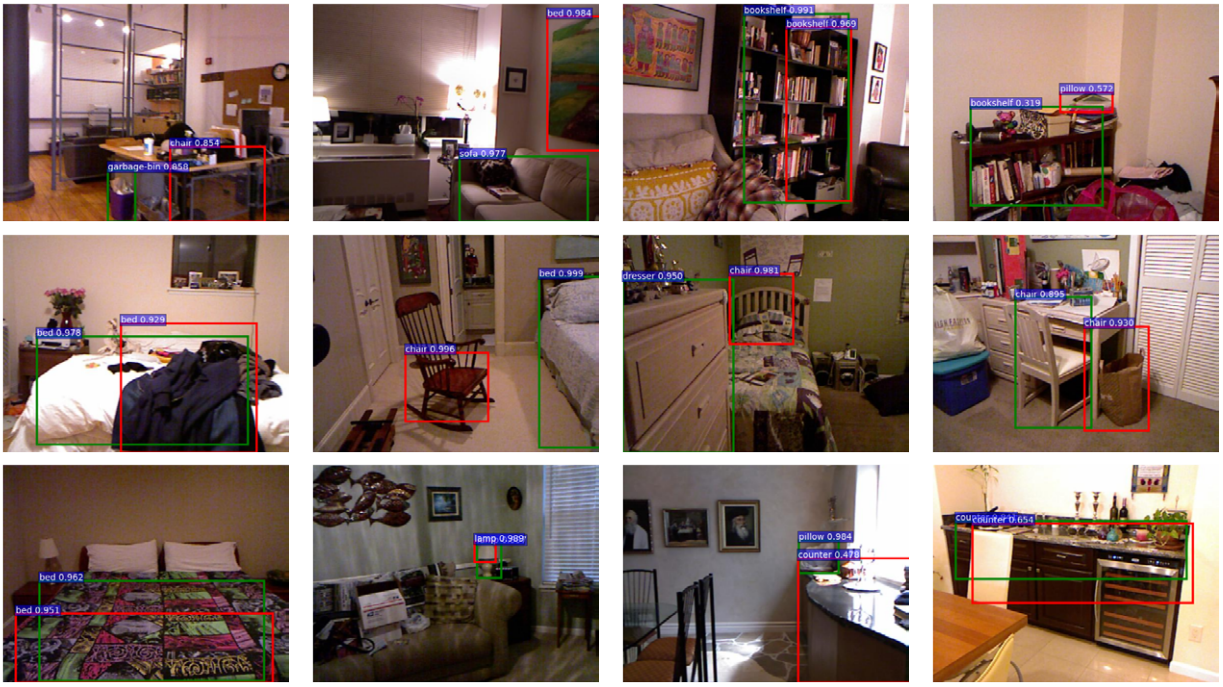


Figure 20.4: Example Detections on the NYUD2 *test set* where our RGB hallucination network's (green box) top scoring detection for the image is correct while the baseline RGB detector's (red box) top scoring detection is incorrect.

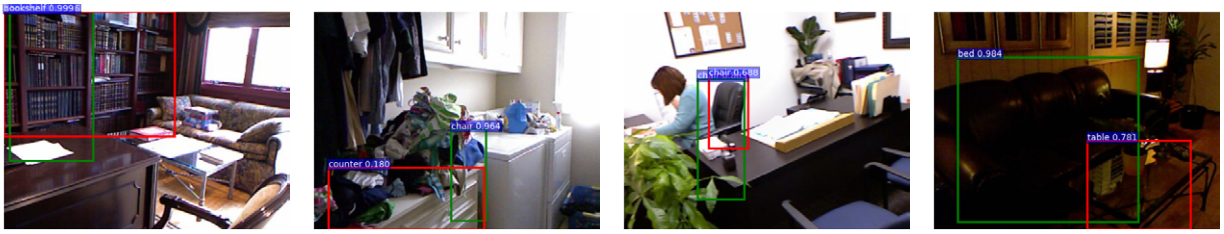


Figure 20.5: Example Detections on the NYUD2 *test set* where our RGB hallucination network's (green box) top scoring detection for the image is a false positive while the baseline RGB detector's (red box) top scoring detection is a true positive.

Chapter 21

Summary

We have presented an algorithm that can transform an RGB object detector into a RGB-D detector which can use depth data at test time to improve performance. Our multi-modal CNN architecture combines mid-level RGB and depth representations to incorporate both modalities into the final object class prediction. This mid-level fusion enables us to train RGB-D detectors without needing complete RGB-D data, unlike most conventional CNN based RGB-D object detection algorithms.

We present experiments showing that our approach provides a 21% relative improvement in performance over just using an RGB detector for categories without no depth data available at training time. We provide insight on how our system helps improve object detection compared to RGB-only detection. Finally, we use our algorithm to adapt the 7.6k category detectors from [85] into a multi-modal RGB-D version, and show qualitative results with this large scale depth detector.

Experiments thus far have been presented using the two stage region proposals and CNN-based feature computation per region, as introduced in R-CNN [59] and Fast R-CNN [60]. Our final goal is to provide a system which can be practically used in a robotics setting. In the future we will work towards making our detectors faster possibly with the use of end-to-end CNN object detection systems like Faster R-CNN [145] and more accurate with use of better CNNs for depth images [69]. We have introduced a novel technique for incorporating additional information, in the form of depth images, at training time to improve our test time RGB only detection models. We accomplish this through our modality hallucination architecture which combines a traditional RGB ConvNet representation with an additional and complementary RGB representation which has been trained to hallucinate depth mid-level features. Our approach outperforms the corresponding Fast R-CNN RGB detection models on the NYUD2 dataset.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "What is an object?" In: *Proc. CVPR*. 2010.

[2] K. Ali and K. Saenko. "Confidence-Rated Multiple Instance Boosting for Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[3] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. "Support vector machines for multiple-instance learning". In: *Proc. NIPS*. 2002, pp. 561–568.

[4] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. "Multiscale Combinatorial Grouping". In: *In: CVPR (2014)*.

[5] Christoph Arndt. *Information Measures: Information and its Description in Science and Engineering*. Signals and Communication Technology. Springer Verlag, 2004, p. 547.

[6] Y. Aytar and A. Zisserman. "Enhancing Exemplar SVMs using Part Level Transfer Regularization". In: *British Machine Vision Conference*. 2012.

[7] Y. Aytar and A. Zisserman. "Tabula Rasa: Model Transfer for Object Category Detection". In: *Proc. ICCV*. 2011.

[8] Jimmy Ba and Rich Caruana. "Do Deep Nets Really Need to be Deep?" In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger. Curran Associates, Inc., 2014, pp. 2654–2662. URL: `http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf`.

[9] S. Basu, M. Bilenko, and R. Mooney. "A Probabilistic Framework for Semi-Supervised Clustering". In: *Proc. 22nd SIGKDD Conference*. 2004.

[10] S. Basu, I. Davidson, and K. Wagstaff, eds. *Constrained Clustering: Advances in Algorithm, Theory, and Applications*. CRC Press, 2008.

[11] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. "Analysis of representations for domain adaptation". In: *Proc. NIPS* (2007).

[12] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. "Review and evaluation of commonly-implemented background subtraction algorithms". In: *Proc. ICPR*. 2008.

[13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538 (2012). URL: http://arxiv.org/abs/1206.5538.

[14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *In Proc. ICML*. 2009.

[15] A. Berg, J. Deng, and L. Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2012. URL: http://www.image-net.org/challenges/LSVRC/2012/.

[16] A. Bergamo and L. Torresani. "Exploiting weakly-labeled Web images to improve object classification: a domain adaptation approach". In: *Neural Information Processing Systems (NIPS)*. Dec. 2010. URL: \url{http://vlg.cs.dartmouth.edu/projects/domainadapt/}.

[17] Hakan Bilen, Vinay P Namboodiri, and Luc J Van Gool. "Object and action classification with latent window parameters". In: *IJCV* 106.3 (2014), pp. 237–251.

[18] J. Blitzer, M. Dredze, and F. Pereira. "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification". In: *ACL* (2007).

[19] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. "Learning bounds for domain adaptation". In: *Proc. NIPS*. 2007.

[20] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. "Integrating structured biological data by Kernel Maximum Mean Discrepancy". In: *Bioinformatics*. 2006.

[21] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih Fu Chang. "Large-scale Visual Sentiment Ontology and Detectors Using Adjective Nown Paiars". In: *ACM Multimedia Conference*. 2013.

[22] John Bridle and Stephen Cox. "RecNorm: Simultaneous normalisation and classification applied to speech recognition". In: *Neural Information Processing Symposium (NIPS)*. 1990.

[23] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. "Signature verification using a Siamese time delay neural network". In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993), pp. 669–688.

[24] Lin Chen, Wen Li, and Dong Xu. "Recognizing RGB Images by Learning from RGB-D Data". In: *CVPR*. 2014.

[25] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. "DLID: Deep Learning for Domain Adaptation by Interpolating between Domains". In: *ICML Workshop on Challenges in Representation Learning*. 2013.

[26] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 539–546.

[27] C. Mario Chrisoudias, Raquel Urtasun, Mathieu Salzmann, and Trevor Darrell. "Learning to Recognize Objects from Unseen Modalities". In: *ECCV*. 2010.

[28] Ramazan Gokberk Cinbis, Jakob Verbeek, Cordelia Schmid, et al. "Multi-fold MIL Training for Weakly Supervised Object Localization". In: *CVPR*. 2014.

[29] A. Coates, A. Karpathy, and A. Ng. "Emergence of Object-Selective Features in Unsupervised Feature Learning". In: *Proc. NIPS*. 2012.

[30] Wenyuan Dai, Yuqiang Chen, Gui-Rong Xue, Qiang Yang, and Yong Yu. "Translated Learning: Transfer Learning across Different Feature Spaces". In: *Proc. NIPS*. 2008.

[31] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection". In: *In Proc. CVPR*. 2005.

[32] H. Daumé III. "Frustratingly easy domain adaptation". In: *ACL*. 2007.

[33] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. "Information-Theoretic Metric Learning". In: *ICML* (2007).

[34] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng. "Large Scale Distributed Deep Networks". In: *Proc. NIPS*. 2012.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR*. 2009.

[36] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. "Weakly supervised localization and learning with generic knowledge". In: *IJCV* (2012).

[37] Tom Diethe, David Roi Hardoon, and John Shawe-Taylor. "Constructing nonlinear discriminants from multiple data views". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 328–343.

[38] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. "Solving the multiple instance problem with axis-parallel rectangles". In: *Artificial intelligence* (1997).

[39] Piotr Dollár and C. Lawrence Zitnick. "Fast Edge Detection Using Structured Forests". In: *CoRR* abs/1406.5549 (2014). URL: `http://arxiv.org/abs/1406.5549`.

[40] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *International Conference in Machine Learning (ICML)*. 2014.

[41] Jeff Donahue, Judy Hoffman, Erik Rodner, Kate Saenko, and Trevor Darrell. "Semi-Supervised Domain Adaptation with Instance Constraints". In: *Computer Vision and Pattern Recognition (CVPR)*. 2013.

[42] M. Dredze and K. Crammer. "Online methods for multi-domain learning and adaptation". In: *Proc. EMNLP*. 2008.

[43] Mark Dredze, Koby Crammer, and Fernando Pereira. "Confidence-Weighted Linear Classification". In: *International Conference on Machine Learning (ICML)*. 2008.

[44] L. Duan, D. Xu, and Ivor W. Tsang. "Learning with Augmented Features for Heterogeneous Domain Adaptation". In: *Proc. ICML*. 2012.

[45] L. Duan, I. W. Tsang, D. Xu, and T. Chua. "Domain adaptation from multiple sources via auxiliary classifiers". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. 2009.

[46] Lixin Duan, Ivor W. Tsang, Dong Xu, and Stephen J. Maybank. "Domain Transfer SVM for Video Concept Detection". In: *CVPR*. 2009.

[47] Lixin Duan, Dong Xu, Ivor Wai-Hung Tsang, and Jiebo Luo. "Visual event recognition in videos by learning from web data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.9 (2012), pp. 1667–1680.

[48] David Eigen and Rob Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture". In: *ICCV*. 2015.

[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". In: *IJCV* 88.2 (June 2010), pp. 303–338.

[50] Ali Farhadi and Mostafa Kamali Tabrizi. "Learning to Recognize Activities from the Wrong View Point". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2008, pp. 154–166.

[51] Jason Farquhar, David Hardoon, Hongying Meng, John S Shawe-taylor, and Sandor Szedmak. "Two view learning: SVM-2K, theory and practice". In: *Advances in neural information processing systems (NIPS)*. 2005, pp. 355–362.

[52] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2010).

[53] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. "Unsupervised Visual Domain Adaptation Using Subspace Alignment". In: *Proc. ICCV*. 2013.

[54] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. "Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT". In: *ArXiv e-prints* abs/1405.5769 (2014). arXiv: `1405.5769`.

[55] K. Fukushima. "A self-organizing neural netowrk model for a mechanism of pattern recognition unaffected by shift position". In: *Biological Cybernetics, 36(4):93-202* (1980).

[56] Carolina Galleguillos, Boris Babenko, Andrew Rabinovich, and Serge Belongie. "Weakly Supervised Object Localization with Stable Segmentations". In: *ECCV*. 2008.

[57] Y. Ganin and V. Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *ICML*. 2015.

[58] Muhammad Ghifary, W. Bastiaan Kleijn, and Mengjie Zhang. "Domain Adaptive Neural Networks for Object Recognition". In: *CoRR* abs/1409.6041 (2014). URL: http://arxiv.org/abs/1409.6041.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *In Proc. CVPR*. 2014.

[60] Ross Girshick. "Fast R-CNN". In: *CVPR*. 2015.

[61] D. Giuliani, R. Gretter, and F. Brugnara. "On-line speaker adaptation on telephony speech data with adaptively trained acoustic models". In: *Proc. ICASSP*. 2009.

[62] Daniel Goehring, Judy Hoffman, Erik Rodner, Kate Saenko, and Trevor Darrell. "Interactive Adaptation of Real-Time Object Detectors". In: *International Conference on Robotics and Automation (ICRA)*. 2014.

[63] B. Gong, Y. Shi, F. Sha, and K. Grauman. "Geodesic Flow Kernel for Unsupervised Domain Adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2066–2073.

[64] R. Gopalan, R. Li, and R. Chellappa. "Domain Adaptation for Object Recognition: An Unsupervised Approach". In: *Proc. ICCV*. 2011.

[65] M. Guillaumin and V. Ferrari. "Large-scale knowledge transfer for object localization in ImageNet". In: *CVPR*. 2012, pp. 3202–3209. DOI: 10.1109/CVPR.2012.6248055.

[66] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. "ImageNet Auto-Annotation with Segmentation Propagation". In: *IJCV* 110 (2014), pp. 328–348. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0713-9. URL: http://dx.doi.org/10.1007/s11263-014-0713-9.

[67] Jian Guo and Stephen Gould. "Deep CNN Ensemble with Data Augmentation for Object Detection". In: *CoRR* abs/1506.07224 (2015). URL: http://arxiv.org/abs/1506.07224.

[68] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. "Perceptual organization and recognition of indoor scenes from rgb-d images". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 564–571.

[69] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. "Cross Modal Distillation for Supervision Transfer". In: *CVPR*. 2016. URL: http://arxiv.org/abs/1507.00448v1.

[70] Saurabh Gupta, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. "Aligning 3D Models to RGB-D Images of Cluttered Scenes". In: *Computer Vision and Pattern Recognition (CVPR)*. 2015.

[71] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. "Learning rich features from rgb-d images for object detection and segmentation". In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 345–360.

[72] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *In Proc. ECCV*. 2014.

[73] James Heckman. "Sample selection bias as a specification error (with an application to estimation of labor supply functions)". In: *National Bureau of Economic Research* (1977).

[74] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning and Representation Learning Workshop*. 2014.

[75] D. Hoeim, Y. Chodpathumwan, and Q. Dai. "Diagnosing Error in Object Detectors". In: *In Proc. ECCV*. 2012.

[76] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. "One-Shot Learning of Supervised Deep Convolutional Models". In: *arXiv 1312.6204; presented at ICLR Workshop*. 2014.

[77] Judy Hoffman, Trevor Darrell, and Kate Saenko. "Continuous Manifold Based Adaptation For Evolving Visual Domains". In: *Computer Vision and Pattern Recognition (CVPR)*. 2014.

[78] Judy Hoffman, Saurabh Gupta, and Trevor Darrell. "Learning with Side Information through Modality Hallucination". In: *In Proc. Computer Vision and Pattern Recognition (CVPR)*. 2016.

[79] Judy Hoffman, Erik Rodner, Jeff Donahue, Brian Kulis, and Kate Saenko. "Asymmetric and Category Invariant Feature Transformations for Domain Adaptation". English. In: *International Journal of Computer Vision* 109.1-2 (2014), pp. 28–41. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0719-3. URL: http://dx.doi.org/10.1007/s11263-014-0719-3.

[80] Judy Hoffman, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell. "Cross-Modal Adaptation for RGB-D Detection". In: *International Conference in Robotics and Automation (ICRA)*. 2016.

[81] Judy Hoffman, Deepak Pathak, Trevor Darrell, and Kate Saenko. "Detector Discovery in the Wild: Joint Multiple Instance and Representation Learning". In: *Computer Vision and Pattern Recognition (CVPR)*. 2015.

[82] Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. "Discovering Latent Domains for Multisource Domain Adaptation". In: *European Conference on Computer Vision (ECCV)*. 2012.

[83] Judy Hoffman, Erik Rodner, Jeff Donahue, Kate Saenko, and Trevor Darrell. "Efficient Learning of Domain-invariant Image Representations". In: *International Conference on Learning Representations*. 2013.

[84] Judy Hoffman, Sergio Guadarrama, Eric Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko. "LSDA: Large Scale Detection through Adaptation". In: *Neural Information Processing Symposium (NIPS)*. 2014.

[85] Judy Hoffman, Sergio Guadarrama, Eric Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko. "LSDA: Large Scale Detection through Adaptation". In: *Neural Information Processing Systems (NIPS)*. 2014.

[86] J. Hosang, R. Benenson, and B. Schiele. "How good are detection proposals, really?" In: *BMVC*. 2014.

[87] Jiayuan Huang, Alexander Smola, Arthur Gretton, Karster Borgwardt, and Bernhard Scholkopf. "Correcting Sample Selection Bias by Unlabeled Data". In: *Neural Information Processing Symposium (NIPS)*. 2006.

[88] D. Hubel and T. Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of Physiology 148 (3): 574-591* (1959).

[89] V. Jain and E. Learned-Miller. "Online domain adaptation of a pre-trained cascade of classifiers". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011.

[90] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. "A Category-Level 3D Object Dataset: Putting the Kinect to Work". In: *Consumer Depth Cameras for Computer Vision*. 2013.

[91] I-Hong Jhuo, Dong Liu, Shih-Fu Chang, and Der-Tsai Lee. "Robust Visual Domain Adaptation with Low-Rank Reconstruction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2168–2175.

[92] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[93] J. Jiang. *A Literature Survey on Domain Adaptation of Statistical Classifiers*. `http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/`. 2008.

[94] J. Jiang and C. X. Zhai. "Instance Weighting for Domain Adaptation in NLP". In: *ACL*. 2007, pp. 264–271.

[95] W. Jiang, E. Zavesky, S. Chang, and A. Loui. "Cross-domain learning methods for high-level visual concept classification". In: *ICIP*. 2008.

[96] R Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* (1960).

[97] Meina Kan, Shiguang Shan, Haihong Zhang, Shihong Lao, and Xilin Chen. "Multi-view discriminant analysis". In: *Proceedings of the European Computer Vision Conference (ECCV)*. Springer, 2012, pp. 808–821.

[98] A. Kembhavi, B. Siddiquie, Roland Miezianko, Scott McCloskey, and L.S. Davis. "Incremental Multiple Kernel Learning for object recognition". In: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009.

[99] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. "Undoing the damage of dataset bias". In: *Proceedings of the 12th European conference on Computer Vision*. 2012.

[100] D. Kifer, S. Ben-David, and J. Gehrke. "Detecting change in data streams". In: *Proc. VLDB*. 2004.

[101] Byung soo Kim, Shili Xu, and Silvio Savarese. "Accurate Localization of 3D Objects from RGB-D Data Using Segmentation Hypotheses". In: *CVPR*. 2013.

[102] Philipp Krähenbühl and Vladlen Koltun. "Geodesic object proposals". In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 725–739.

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proc. NIPS*. 2012.

[104] B. Kulis, K. Saenko, and T. Darrell. "What You Saw is Not What You Get: Domain Adaptation Using Asymmetric Kernel Transforms". In: *Proc. CVPR*. 2011.

[105] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. "Semi-supervised Graph Clustering: A Kernel Approach". In: *Proc. ICML*. 2005.

[106] Brian Kulis and Michael I. Jordan. "Revisiting k-means: New Algorithms via Bayesian Nonparametrics". In: *ICML*. Arxiv:1111.0352. 2012.

[107] M Pawan Kumar, Benjamin Packer, and Daphne Koller. "Self-paced learning for latent variable models". In: *In Proc. NIPS*. 2010.

[108] Kevin Lai, Liefeng Bo, and Dieter Fox. "Unsupervised Feature Learning for 3D Scene Labeling". In: *IEEE International Conference on on Robotics and Automation*. 2014.

[109] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. "A large-scale hierarchical multi-view RGB-D object dataset". In: *ICRA*. 2011.

[110] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories". In: *Computer Vision and Pattern Recognition (CVPR)*. 2006.

[111] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* (1989).

[112] Y. J. Lee, A. Efros, and M. Hebert. "Style-aware Mid-level Representation for Discovering Visual Connections in Space and Time". In: *Proc. ICCV*. 2013.

[113] K. Lenc and A. Vedaldi. "Understanding image representations by measuring their equivariance and equivalence". In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[114] A. Levey and gM. Lindenbaum. "Sequential Karhunen-Loeve basis extraction and its application to images". In: *Image Processing, IEEE Transactions on* (2000).

[115] Ruonan Li and Todd Zickler. "Discriminative virtual views for cross-view action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2855–2862.

[116] X. Li. "Regularized Adaptation: Theory, Algorithms and Applications". In: *PhD thesis, University of Washington, USA*. 2007.

[117] X. Li, K. Wang, W. Wang, and Y. Li. "A multiple object tracking method using Kalman filter". In: *Information and Automation (ICIA), 2010 IEEE International Conference on*. 2010.

[118] Hank Liao. "Speaker Adaptation of Context Dependent Deep Neural Networks". In: *ICASSP*. 2013.

[119] Dahua Lin, Sanja Fidler, and Raquel Urtasun. "Holistic scene understanding for 3D object detection with RGBD cameras". In: *ICCV*. 2013.

[120] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Pitro Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common objects in context". In: *arXiv:1405.0312 [cs.CV]* (2014).

[121] Fayao Liu, Chunhua Shen, and Guosheng Lin. "Deep Convolutional Neural Fields for Depth Estimation from a Single Image". In: *CVPR*. 2015. URL: `http://arxiv.org/abs/1411.6387`.

[122] Wei Liu, Andrew Rabinovich, and Alexander C Berg. "Parsenet: Looking wider to see better". In: *arXiv preprint* (2015).

[123] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CVPR* (Nov. 2015). arXiv: `1411.4038 [cs.CV]`.

[124] Mingsheng Long and Jianmin Wang. "Learning Transferable Features with Deep Adaptation Networks". In: *ICML*. 2015.

[125] D. G. Lowe. "Distinctive image features from scale-invariant key points". In: *IJCV* (2004).

[126] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. "Domain Adaptation: Learning Bounds and Algorithms". In: *COLT*. 2009.

[127] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. "Domain Adaptation with Multiple Sources". In: *NIPS*. 2008.

[128] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. "Multiple Source Adaptation and the Rényi Divergence". In: *UAI*. 2009, pp. 367–374.

[129] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker. "Multidigit recognition using a space displacement neural network". In: *NIPS*. 1991, pp. 488–495.

[130] Ofer Matan, Christopher J.C. Burges, Yann Le Cun, and John S. Denker. "Multi-Digit Recognition Using A Space Displacement Neural Network". In: *Neural Information Processing Systems*. Morgan Kaufmann, 1992, pp. 488–495.

[131] M. Minsky and S. Papert. "Perceptrons". In: *Cambridge, MA: MIT Press* (1969).

[132] Damian Mrowca, Marcus Rohrbach, Judy Hoffman, Ronghang Hu, Kate Saenko, and Trevor Darrell. "Spatial Semantic Regularisation for Large Scale Object Detection". In: *ICCV*. 2015.

[133] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. "Domain Generalization via Invariant Feature Representation". In: *Proceedings of ICML*. 2013, pp. 10–18.

[134] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. "Multimodal deep learning". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 689–696.

[135] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. "Toward automatic phenotyping of developing embryos from videos". In: *IEEE Transactions on Image Processing*. 2005, 14(9):1360–1371.

[136] Wanli Ouyang, Ping Luo, Xingyu Zeng, Shi Qiu, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Yuanjun Xiong, Chen Qian, et al. "DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection". In: *arXiv:1409.3505* (2014).

[137] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on Knowledge and Data Engineering*. 2010.

[138] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. "Domain Adaptation via Transfer Component Analysis". In: *IJCA*. 2009.

[139] Megha Pandey and Svetlana Lazebnik. "Scene recognition and weakly supervised object localization with deformable part-based models". In: *Proc. ICCV*. 2011.

[140] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. "Weakly-and semi-supervised learning of a dcnn for semantic image segmentation". In: *CoRR* abs/1502.02734 (2015).

[141] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. "Constrained Convolutional Neural Networks for Segmentation". In: *ICCV*. 2015.

[142] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Multi-Class Multiple Instance Learning". In: *CoRR* abs/1412.7144 (2014). URL: `http://arxiv.org/abs/1412.7144`.

[143] Novi Quadrianto and Christoph H. Lampert. "Learning multi-view neighborhood preserving projections". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2011, pp. 425–432.

[144] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[145] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015.

[146] Q. Rentmeesters, P-A Absil, P. Van Dooren, K. Gallivan, and A. Srivastava. "An efficient particle filtering technique on the Grassmann manifold". In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. 2010.

[147] Alfréd Rényi. "On Measures of Entropy and Information". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 1961, pp. 547–561.

[148] Erik Rodner, Judy Hoffman, Jeff Donahue, Trevor Darrell, and Kate Saenko. "Towards Adapting ImageNet to Reality: Scalable Domain Adaptation with Implicit Low-rank Transformations". In: *arXiv preprint arXiv:1308.4200* (2013).

[149] Frank Rosenblatt. "The Perceptron – a perceiving an drecognizing automaton". In: *Report 850460-1, Cornell Aeronautical Laboratory* (1957).

[150] D. Ross, J. Lim, and M.H. Yang. "Adaptive Probabilistic Visual Tracking with Incremental Subspace Update". In: *European Conference on Computer Vision (ECCV)*. 2004.

[151] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. "Adapting Visual Category Models to New Domains". In: *Proc. ECCV*. 2010.

[152] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *CoRR* abs/1312.6229 (2013).

[153] S. Shalev-Shwartz. "Online Learning and Online Convex Optimization". In: *Found. Trends Mach. Learn.* (2012).

[154] Abhishek Sharma, Abhishek Kumar, H Daume, and David W Jacobs. "Generalized multiview analysis: A discriminative latent space". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 2160–2167.

[155] V. Sharmanska, N. Quadrianto, and C.H. Lampert. "Learning to Rank Using Privileged Information". In: *ICCV*. 2013, pp. 825–832. DOI: 10.1109/ICCV.2013.107.

[156] Abhinav Shrivastava and Abhinav Gupta. "Building Part-based Object Detectors via 3D Geometry". In: *ICCV*. 2013.

[157] C. Siagian and L. Itti. "Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007).

[158] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *ECCV*. 2012.

[159] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014).

[160] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. "Unsupervised discovery of mid-level discriminative patches". In: *ECCV*. 2012.

[161] Parthipan Siva, Chris Russell, and Tao Xiang. "In defence of negative mining for annotating weakly labelled data". In: *ECCV*. 2012.

[162] Parthipan Siva, Chris Russell, Tao Xiang, and Lourdes Agapito. "Looking beyond the image: Unsupervised learning for object saliency and detection". In: *Proc. CVPR*. 2013.

[163] H. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. "On learning to localize objects with minimal supervision". In: *ICML*. 2014.

[164] Hyun Oh Song, Yong Jae Lee, Stefanie Jegelka, and Trevor Darrell. "Weakly-supervised Discovery of Visual Pattern Configurations". In: *Proc. NIPS*. 2014.

[165] Shuran Song and Jianxiong Xiao. "Sliding shapes for 3D object detection in depth images". In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 634–651.

[166] Luciano Spinello and Kai O. Arras. "Leveraging RGB-D Data: Adaptive Fusion and Domain Adaptation for Object Detection". In: *ICRA*. 2012.

[167] Nitish Srivastava and Ruslan R Salakhutdinov. "Multimodal learning with deep boltzmann machines". In: *Advances in neural information processing systems*. 2012, pp. 2222–2230.

[168] C. Stauffer and W. E L Grimson. "Adaptive background mixture models for real-time tracking". In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* 1999.

[169] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. "Unstructured human activity detection from rgbd images". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 842–849.

[170] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions". In: *arXiv:1409.4842* (2014).

[171] Jie Tang, Stephen Miller, Arjun Singh, and Pieter Abbeel. "A Textured Object Recognition Pipeline for Color and Depth Image Data". In: *International Conference on Robotics and Automation*. 2012.

[172] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. "Histogram of Oriented Normal Vectors for Object Recognition with a Depth Sensor". In: *ACCV*. 2012.

[173] Ekaterina Taralova, Fernando De la Torre Frade, and Martial Hebert. "Source Constrained Clustering". In: *13th International Conference on Computer Vision 2011*. 2011.

[174] T. Tommasi, T. Tuytelaars, and B. Caputo. "A Testbed for Cross-Dataset Analysis". In: *TASK-CV Workshop, ECCV*. 2014.

[175] A. Torralba and A. Efros. "Unbiased Look at Dataset Bias". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 1521–1528.

[176] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. "Deep Domain Confusion: Maximizing for Domain Invariance". In: *CoRR* abs/1412.3474 (2014). URL: http://arxiv.org/abs/1412.3474.

[177] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. "Simultaneous Deep Transfer Across Domains and Tasks". In: *International Conference in Computer Vision (ICCV)*. 2015.

[178] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. "Selective Search for Object Recognition". In: *IJCV* 104.2 (2013), pp. 154–171.

[179] Vladimir Vapnik and Akshay Vashist. "A new learning paradigm: Learning using privileged information". In: *Neural Networks* 22.5â6 (2009). Advances in Neural Networks Research: {IJCNN20092009} International Joint Conference on Neural Networks, pp. 544 –557. ISSN: 0893-6080. DOI: http://dx.doi.org/10.1016/j.neunet.2009.06.042. URL: http://www.sciencedirect.com/science/article/pii/S0893608009001130.

[180] Alexander Vezhnevets and Vittorio Ferrari. "Associative Embeddings for Large-scale Knowledge Transfer with Self-assessment". In: *CVPR* (2014).

[181] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. "Constrained k-means clustering with background knowledge". In: *Proc. ICML*. 2001.

[182] A. Wang, J. Lu, J. Cai, T. Cham, and G. Wang. "Large-Margin Multi-Modal Deep Learning for RGB-D Object Recognition". In: *IEEE Transactions on Multimedia*. 2015.

[183] Chong Wang, Weiqiang Ren, Kaiqi Huang, and Tieniu Tan. "Weakly Supervised Object Localization with Latet Category Learning". In: *European Conference on Computer Vision (ECCV)*. 2014.

[184] Xiaolong Wang and Abhinav Gupta. "Unsupervised Learning of Visual Representations using Videos". In: *ICCV*. 2015. URL: http://arxiv.org/abs/1505.00687.

[185] Ralph Wolf and John C. Platt. "Postal address block location using a convolutional locator network". In: *in Advances in Neural Information Processing Systems 6*. Morgan Kaufmann Publishers, 1994, pp. 745–752.

[186] J. Xu, S. Ramos, D. Vázquez, and A.M. López. "Domain Adaptation of Deformable Part-Based Models". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* In Press (2014).

[187] Zheng Xu, Wen Li, Li Niu, and Dong Xu. "Exploiting Low-Rank Structure from Latent Domains for Domain Generalization". In: *European Conference in Computer Vision (ECCV)*. 2014.

[188] J. Yang, R. Yan, and A. Hauptmann. "Adapting svm classifiers to data with shifted distributions". In: *ICDM Workshops*. 2007.

[189] J. Yang, R. Yan, and A. G. Hauptmann. "Cross-domain video concept detection using Adaptive SVMs". In: *ACM Multimedia* (2007).

[190] Edmund Shanming Ye. "Object Detection in RGB-D Indoor Scenes". MA thesis. EECS Department, University of California, Berkeley, 2013. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-3.html`.

[191] Chun-Nam John Yu and Thorsten Joachims. "Learning structural svms with latent variables". In: *Proc. ICML*. 2009, pp. 1169–1176.

[192] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *ICLR* (2016).

[193] Alan L Yuille and Anand Rangarajan. "The concave-convex procedure". In: *Neural Computation* 15.4 (2003), pp. 915–936.

[194] M. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *ArXiv e-prints* (2013). arXiv: `1311.2901`.

[195] Cha Zhang, John C Platt, and Paul A Viola. "Multiple instance boosting for object detection". In: *Advances in neural information processing systems*. 2005.

[196] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. "Conditional Random Fields as Recurrent Neural Networks". In: *ICCV*. 2015.

[197] C Lawrence Zitnick and Piotr Dollár. "Edge boxes: Locating object proposals from edges". In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 391–405.