ABSTRACT

Title of dissertation: UNDERSTANDING OBJECTS
IN THE VISUAL WORLD

Ejaz Ahmed, Doctor of Philosophy, 2015

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

One way to understand the visual world is by reasoning about the objects present in it: their type, their location, their similarities, their layout etc. Despite several successes, detailed recognition remains a challenging tasks for current computer vision systems. This dissertation focuses on building systems that improve on the state-of-the-art on several fronts. On one hand, we propose better representations of visual categories that enable more accurate reasoning about their properties. To learn such representations, we employ machine learning methods that leverage the power of big-data. On the other hand, we present solutions to make current frameworks more efficient without losing on performance.

The first part of the dissertation focuses on improvements in efficiency. We first introduce a fast automated mechanism for selecting a diverse set of discriminative filters and show that one can efficiently learn a universal model of filter "goodness" based on properties of the filter itself. As an alternative to the expensive evaluation of filters, which is often the bottleneck in many techniques, our method has the potential of dramatically altering the trade-off between the accuracy of a filter based method and the cost of training. Second, we present a method for linear dimensionality reduction which we call composite discriminant factor analysis (CDF). CDF searches for a discriminative but compact feature subspace in which the classifiers can be trained, leading to an order of magnitude saving in detection time.

In the second part, we focus on the problem of person re-identification, an important component of surveillance systems. We present a deep learning architecture that simultaneously learns features and computes their corresponding similarity metric. Given a pair of images as input, our network outputs a similarity value indicating whether the two input images depict the same person. We propose new layers which capture local relationships among mid-level features, produce a high-level summary of these relationships and spatially integrate them to give a holistic representation.

In the final part, we present a semantic object selection framework that uses natural language input to perform image editing. In the general context of interactive object segmentation, many of the methods that utilize user input (such as mouse clicks and mouse strokes) often require significant user intervention. In this work, we present a system with a far simpler input method: the user only needs to give the name of the desired object. For this problem we present a solution which borrows ideas from image retrieval, segmentation propagation, object localization and convolution neural networks.

UNDERSTANDING OBJECTS IN THE VISUAL WORLD

by

Ejaz Ahmed

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor David W. Jacobs
Professor Yiannis Aloimonos
Professor Ramani Duraiswami
Professor Jimmy Lin

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Larry Davis for providing me valuable guidance, giving me the freedom to work on different types of problems and to freely explore my research ideas.

During my Ph.D. I got the opportunity to work at various places in academia and industry. The work done at these places has played a pivotal role in my thesis. I would like to thank my mentors at Adobe Research, Dr. Scott Cohen and Dr. Brian Price, for their guidance on the problem of semantic object selection. I would like to express my gratitude to Professor Gregory Shakhnarovich and Professor Subhransu Maji with whom I worked at TTI-Chicago on filter selection. I had meaningful discussions with Dr. Michael Jones and Dr. Tim Marks at MERL on the problem of person re-identification. Finally, I would like to thank Dr. Vlad Morariu who helped me during the initial phase of my Ph.D.

I am grateful for the support that my family has provided throughout the ups and downs of the Ph.D. process. My parents, Mushtaq and Saira, have always understood the seriousness of Ph.D. workload. They have always sheltered me from various family responsibilities. My father has filled me with positivity whenever I talked to him. I would specially like to thank my long time girl friend and soon to be wife, Neha, for the encouragement that she has provided. She has patiently listened to me recount my successes and failures and always motivated me whenever I was low-spirited.

My friends and colleagues have made my graduate experience memorable and fun

through discussions. I have had numerous conversations with them over which we used to discuss anything and everything of life. I would like to thank my UMD cricket teammates for organizing regular practice sessions and games. Cricket provided a balance between academics and sports which helped me relieve stress.

Finally, I thank my committee members for taking time out of their busy schedules to make the final step of my doctorate possible.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Automatic methods for detailed understanding of images are essential for various applications in diverse domains such as, postal service, retail, medicine, robotics, automotive safety, surveillance, entertainment, marketing, etc. One of the ways to understand the visual world is by reasoning about the objects present in it. One might try to answer questions like, what type of objects are present in the image, where are they located, how are they related to each other, what are their exact outline and 3D geometry etc. By finding answers to these questions, one might infer about the scene. For instance, given an image of a crossroad, we can start by finding objects like pedestrians and cars in the image. We can also find their relative positions and then infer whether the car should go ahead or wait for the pedestrian to cross the road. A typical surveillance scenario arises when there are multiple cameras installed at airports covering different areas of the airport and one might want to track and re-identify people as they move from one camera to the other. Another example is image editing tools, such as Photoshop, which are used to modify objects in appearance to improve visual quality of the image.

Most of these questions are formulated as various computer vision problems like object detection, object recognition, object segmentation, semantic segmentation, person re-identification, tracking etc. Many times the problem at hand is not well defined in the

existing literature and one might have to propose their own instantiation of the problem. Despite several successes, detailed recognition is beyond the current computer vision systems. This is a challenging task, and to make progress we have to make advances on several fronts. We need better representations of visual categories that can enable accurate reasoning about their properties, as well as machine learning methods that can leverage big-data to learn such representations. For real time systems we need to make current frameworks more efficient without losing on performance. This work focuses on building such systems improving the state-of-the-art in terms of accuracy and efficiency, and proposing new vision problems with wide applicability.

This work introduces the following contributions to the task of object understanding: 1) an efficient automatic mechanism for selecting a diverse set of discriminative filters for object detection, 2) a deep learning architecture for simultaneously learning features and a corresponding similarity metric for the problem of person re-identification, 3) a semantic object selection framework which advances image editing via natural language input, and 4) a linear dimensionality reduction method, composite discriminant factor (CDF) analysis, which searches for a discriminative but compact feature subspace.

1.1 Efficient Object Detectors

Object detection is one of the fundamental problems of computer vision. It has a wide range of applications and a good detection system can benefit many other related tasks such as, object retrieval, object segmentation, tracking etc. In this work we tackle the problem of vehicle detection from aerial images and videos captured from satellite

cameras and drones. This is a challenging problem and conventional object detectors, part based or even deep learning based, are unlikely to be successful since target objects are quite small and many times in extremely low resolution. We solve this by extracting very high-dimensional features to represent the object. This makes the number of training samples to be much smaller than the number of features and the features have high degree of multi-collinearity. Moreover, developing efficient classifiers for object detection is vital since object detection has the complexity of searching over various locations, orientations and scales in an image to find an object.

We have developed a linear dimensionality reduction method called Composite Discriminant Factor (CDF) analysis, which searches for a discriminative but compact feature subspace. Classifiers can be trained in this compact feature space, so as to avoid the problems of multicollinearity arising in a high dimensional feature space. This offers better trade-offs between representation power and efficiency, leading to a compact representation and orders of magnitude saving in detection time. This class-aware dimensionality reduction method has scope beyond object detection, our experiments with various machine learning data sets and multi-class action recognition suggest that CDF is a good alternative to linear SVM for many state-of-the-art vision and non-vision tasks. We have also compiled and released a large scale multi-class vehicle detection data set.

One of the common ways to approach object detection, or object recognition in general, is to represent a visual category as collection of filters. These collections are used for various vision tasks such as fine-grained classification, pose estimation, segmentation and detection. The success of these methods relies on how these filters are selected. We developed an automatic mechanism for selecting a diverse set of discriminative filters.

As an alternative to the expensive explicit evaluation that is often the bottleneck in many methods, such as poselets, this has the potential to dramatically alter the tradeoff between accuracy of a part based model and the cost of training. Our filter selection method allows us to quickly discard filters that are not promising, allowing us to improve its detection performance while speeding up training by an order of magnitude. This work suggests that it is possible to evaluate the discriminative quality of a set of filters based purely on their intrinsic properties. Beyond direct savings in training time for filter-based models, this evaluation may lead to speeding up filter-based detection methods at test time, when used as an attention mechanism to reduce number of convolutions and/or hashing lookups. We have also built an interactive framework for poselet selection to show that humans can help in building better detectors by including their knowledge of good filters.

1.2 Deep Learning Architecture for Person Re-Identification

Deep convolutional neural networks have recently demonstrated excellent results on various vision problems. This has been possible because of the availability of large amount of visual information for various tasks and because of the increase in the compute capabilities of machines. In this work we propose a novel deep convolutional neural network for the problem of person re-identification. Person re-identification is the problem of identifying people in images that have been taken across different cameras, or across time using a single camera. Re-identification is an important capability for surveillance systems as well as human-computer interaction systems. It is an especially difficult problem, because large variations in viewpoint and lighting across different views can cause

two images of the same person to look quite different and can cause images of different people to look very similar. Our architecture outperforms the state-of-the-art on a large person re-identification data set (CUHK03) by a significant margin. We also show that models learned by our method on a large data set can be adapted to new, smaller data sets as well. We demonstrate this by evaluating our method on small data sets (CUHK01 and VIPeR). Apart from significant improvements in performance our method gives insight into questions like 'What parts are most useful for person matching?'. Variation of this architecture can be used for other tasks, for example retrieving similar dresses from an online clothing store.

1.3 Semantic Object Selection for Image Editing

With the current proliferation of natural language interfaces for all sorts of tasks (e.g. Siri, PixelTone), there is a need for advancing image editing via natural language input. In PixelTone (a multimodal interface for image editing) a user can make request such as "Make the *cat* brighter". Since the system does not know which pixels are cat pixels, the user has to paint on the image to mark the cat, name the selection, and then the user can issue semantic editing requests that mention "cat". One of the primary operations in image editing is object selection. Many interactive object selection methods have been proposed which utilize user input in the form of mouse clicks and mouse strokes, and often require a lot of user intervention.

To this end, we introduced the new problem of Semantic Object Selection in which a user simply specifies the class of the object to be selected in an image. This results in

a system with a far simpler input method for interactive object segmentation and can be used for image editing tasks. For example, in case of PixelTone requests, our method can be used to identify the cat pixels to be made brighter without any further input from the user.

We propose a solution to this problem that scales well with the number of classes. In order to solve this problem we propose an exemplar-based localization method which relies on object retrieval. We break the image into object proposals and validate the presence of the object in the proposal. Location priors obtained in this way are then used to get an image specific appearance model and both are used to solve the segmentation problem in an MRF framework.

In order for this system to be practical, we need to design innovative architectures, both by developing novel parts and by leveraging existing components. Existing components such as, object retrieval, object classification, localization, saliency estimation, energy minimization etc. should be improved in order to make the system efficient and accurate. In order to improve image editing experience, we should go beyond object selection and exploit semantics from user queries. For example user should be able to issue more complex queries like, "make the dog next to the tree brighter" or "select the brown dog" to disambiguate among multiple dogs.

1.4 Organization

The dissertation is organized as follows. In Chapter 2 we present interactive method for filter selection. We introduce the concept of collection of filters and show that humans

can help build better detectors by including their knowledge of good filters. In Chapter 3 we extend the idea of filter selection and present an automatic and efficient method to select diverse set of discriminative filters. In Chapter 4 we present a linear dimensionality reduction method, Composite Discriminant Factor (CDF) analysis, which searches for a discriminative but compact feature subspace. In Chapter 5 we present a deep learning architecture for simultaneously learning features and a corresponding similarity metric for the problem of person re-identification. In Chapter 6 we present an object selection system with a far simpler input method than current systems. We conclude the dissertation in Chapter 7 and give directions for future work.

Chapter 2: Using Human Knowledge to Judge Filter Goodness: Interactive Filter Selection

It is a common practice to model object detectors as collection of filters. For these detectors to be effective, it is important to select "good filters" covering most of the variation of the data. In order to achieve this, these methods invest majority of their time selecting a good subset of filters from a large pool. Good filters are less cluttered and their gradients are spatially correlated. Humans can differentiate between a good filter and a bad one by visualizing them. In addition humans bring with them the knowledge of diversity and effectiveness of filters, properties which are difficult to model. In this work, we show that humans can help build better detectors by including their knowledge of good filters. We show this by building an interactive framework for poselet selection. Our interactive framework improves the detection performance on the PASCAL VOC dataset and significantly improves the training time. This work has been published in [7].

2.1 Introduction and Motivation

A common approach to modeling a visual category is to represent it as a composition of smaller fragments (parts) arranged in a variety of layouts or as library of exemplars, more generally, as collection of filters. Modeling a category as collection of filters helps

	aplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
Oracle	**32.37**	50.00	**12.82**	16.36	**31.57**	41.30	**56.00**	**20.84**	**19.20**	37.55
Interactive	29.84	**50.88**	12.57	**20.16**	31.48	**43.59**	55.82	19.85	18.29	**40.08**
Overlap	23	20	31	19	30	31	29	30	20	33
	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Oracle	14.51	**17.04**	37.63	**35.91**	**36.65**	13.14	**31.87**	23.35	24.31	28.21
Interactive	**15.42**	16.66	**44.47**	35.08	35.56	**13.26**	31.55	**27.03**	**25.50**	**30.66**
Overlap	16	41	26	33	45	22	17	2	22	14

Table 2.1: Per Category Results on PASCAL VOC 2007. Best results are highlighted in bold. Best mean average precision of 29.86 is obtained using our interactive method vs 29.03 for oracle. Note that this method resulted in significant speed up of the selection pipeline (5-10mins as compared against 15Hrs). Last row (Overlap) shows number of filters common to the two methods for each category. On average there are around 25 filters common to the two methods.

in modeling a large amount of variation in data. In this work our focus is on an architecture which has two major steps (i) Candidate Generation and (ii) Selection. Examples of such architectures are poselets [8, 9], exemplar SVM [10] and discriminative patches [11]. In this paper we investigate poselets.

A common approach to modeling a visual category is to represent it as a composition of smaller fragments (parts) arranged in a variety of layouts or as library of exemplars, more generally, as collection of filters. Modeling a category as collection of filters helps in modeling a large amount of variation in data. In this work our focus is on an architecture which has two major steps (i) Candidate Generation and (ii) Selection. Examples of such architectures are poselets [8, 9], exemplar SVM [10] and discriminative patches [11]. In this paper we investigate poselets.

Candidate generation step involves training many HOG detectors [12] each representing a part of the object. This step involves training a HOG detector and mining for hard negatives, and is moderately expensive. The Selection step is most expensive and involves evaluating each generated part filter (large pool) on a large number of positive

and negative examples. The selected parts should be (i) Discriminative - they should fire only at meaningful locations on the test image and (ii) Diverse - many candidate parts are highly similar to each other, there is no point in selecting very similar parts twice.

What is a good part? One way to determine the discriminativeness of filters is by evaluating them on a held-out set [9]. This is very expensive as there are many candidate parts. However, good discriminative parts have the property that they are less cluttered and their gradients are spatially correlated. Figure 2.1 shows examples of good and bad filters. It is easy to tell the difference between the two by visual inspection. For good filters, neighboring gradient orientation bins are active simultaneously and majority of them are entirely suppressed. This is due to the fact that the template has to account for small variations in local gradient directions in order to be robust. Also note that if a certain gradient orientation is encouraged, its orthogonal counterpart is often penalized. Also, dominant orientation bins of neighboring cells tend to coincide forming line segments or disagree by an angle to form curves and corners, or be parallel. This could be attributed to the fact that the template has to be robust to small spatial variations in alignment of training samples, which results in neighboring cell show similar patterns. Also, gradient based nature of HOG features tend to capture object outlines which are often smooth lines and curves. [13] exploited these properties in a generative framework to come up with structured prior for the SVM loss function. In addition to these there are certain properties which are difficult to model. For instance, a filter trained on the legs of a person (parallel lines) is although a good one, in the sense that gradients might be less cluttered. But in practice such a filter will not be effective since parallel lines are common in the real world and such a detector will fire all over the place.

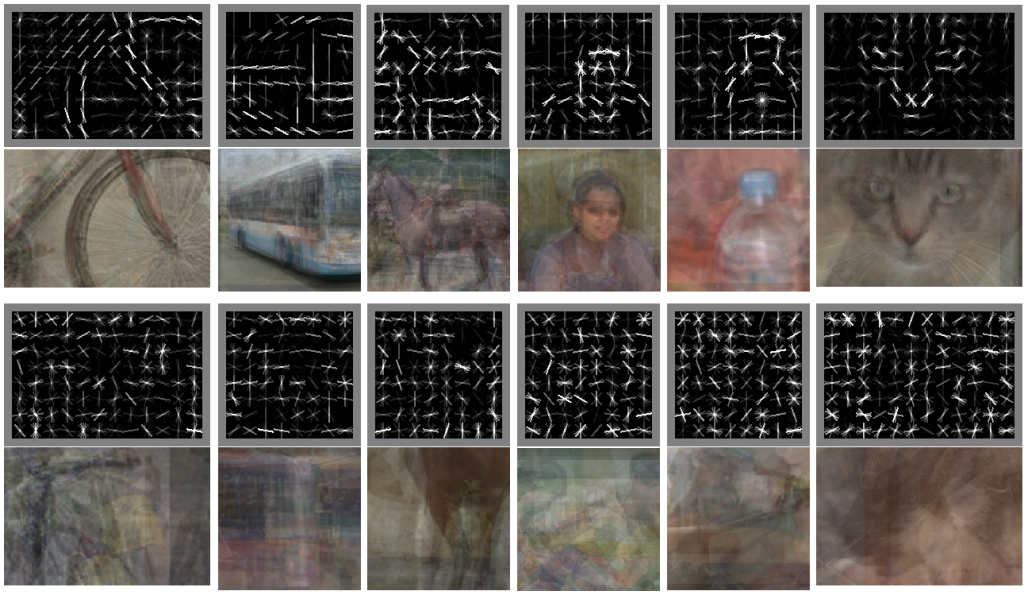

Figure 2.1: Good-Bad Filters and Seed Image(made by averaging top 10 seeds): Top 2 rows show good filters and bottom 2 rows show bad filters. Categories from left to right are bicycle, bus, horse, person, bottle and cat. Note the difference between gradient orientations for good and bad filters. Also seed images for the good filters are cleaner.

In this work we show that it is possible to select filters with the help of human knowledge of good filters. Humans bring with them the knowledge about (i) good filters - by visualizing one can tell the difference between good and bad filters, (ii) diverse filters - humans know if two filters represent the same part and (iii) effective filters (leg example). We present an interactive framework to select discriminative and diverse set of filters with minimal effort.

2.2 Original Poselet Selection

In the original paper [9] selection is performed by evaluating candidate poselets on the entire training set, and a subset is selected using a greed coverage algorithm that iteratively picks poselets that offer highest increase in detection accuracy at a fixed false

positive rate. This is time consuming and takes 76% (15Hrs) of the total time (20Hrs) to train. Also note that this is just an approximation to discriminative and diversity criteria.

2.3 Interactive Selection

The idea is to display filters to a user and let it select good and diverse set of filters. To assist the user a simple heuristic of filter quality (norm) is used. As we know selected filters should be diverse too, hence we define a measure for similarity between a non selected filter and a set of selected filters. Initially, we display the filters sorted according to the descending value of normalized norm ($\frac{norm}{\#cells}$) of the filter. This acts as a heuristic of good filter and tends to push good filters higher in the visualization box. The user then browses through the filters and selects $k = 5$-10 filters. Then re-ranking of non-selected filters is done according to their similarity to the selected filters. For obtaining the similarity we compute the bounding box overlap of top $r = 3\%$ of the ordered list of training examples of poselets. Similarity of a non-selected filter with a set of selected filters is determined as k-th order value of similarity between candidate part and those already selected. This helps assist user to select good filters with a minimal effort and without browsing through all of the candidate filters. The process is then repeated until desired number of poselets are selected.

2.4 Experiments

We constructed a poselet model by selecting 100 poselets from a set of 800 poselets using original poselet selection method (Oracle) and using our interactive framework.

For our interactive framework, initially user selects 10 poselets and then clicks "process of diversity". In the subsequent iterations user selects 5-10 poselets before clicking for "process of diversity". These steps are repeated until 100 poselets are selected. It takes about 5-10mins for selecting 100 poselets for a category. We evaluate these models as detectors on PASCAL VOC 2007 dataset. To isolate the effect of poselet selection, we use a simplified implementation that avoids some of the post-processing steps (Q-poselet models). The results are reported in table 2.1. We achieve an improvement in the detection performance, $\Delta(mAP) = +0.86$.

2.5 Conclusion

We have presented an interactive framework for selecting discriminative and diverse set of parts. With our method user knowledge of a good part enters the training pipeline. Our method significantly improves the training time, it takes about 5-10mins for a user to interactively select 100 poselets. As compared to 15Hrs (76% of 20Hrs) for computer to select them. Last but not the least, our method helps in constructing better detectors ($\Delta(mAP) = +0.86$). Moreover this method helps us to understand what a good detector is and gives direction for developing automatic methods for good part selection.

Chapter 3: Knowing a good HOG filter when you see it: Efficient selection of filters for detection

In this chapter we extend the idea of filter selection and present an automatic and efficient method to select diverse set of discriminative filters. Collections of filters based on histograms of oriented gradients (HOG) are common for several detection methods, notably, poselets and exemplar SVMs. The main bottleneck in training such systems is the selection of a subset of good filters from a large number of possible choices. We show that one can learn a universal model of part "goodness" based on properties that can be computed from the filter itself. The intuition is that good filters across categories exhibit common traits such as, low clutter and gradients that are spatially correlated. This allows us to quickly discard filters that are not promising thereby speeding up the training procedure. Applied to training the poselet model, our automated selection procedure allows us to improve its detection performance on the PASCAL VOC data sets, while speeding up training by an *order of magnitude*. Similar results are reported for exemplar SVMs. This work has been published in [14].

3.1 Introduction

A common approach to modeling a visual category is to represent it as a mixture of appearance models. These mixtures could be part-based, such as those in poselets [8, 9], and deformable part-based models [15], or defined globally, such as those in exemplar SVMs [10]. Histograms of oriented gradient (HOG) [12] features are often used to model the appearance of a single component of these mixtures. Details on how these mixture components are defined, and discovered, vary across methods; in this paper our focus is on a common architecture where a pool of candidate HOG filters is generated from instances of the category, and perhaps some negative examples, followed by a selection stage in which filters are, often in a greedy fashion, selected based on their incremental contribution to the detection performance.

The candidate generation step is, typically, at most moderately expensive. The selection stage, however, requires an expensive process of evaluating each candidate on a large set of positive and negative examples. There are two sources of inefficiency in this: (i) **Redundancy**, as many of the candidates are highly similar to each other, since the generation process is driven by frequency of keypoint configurations for poselets, of examples for exemplar SVMs; (ii) **Noise**, as many of the candidates are not discriminative, not localizable (e.g., due to aperture effect) or not repeatable.

In this paper we address both of these inefficiencies, and propose a method that automatically selects from a large pool of filters generated for a category, a small subset that is likely to contain non-redundant discriminative ones. We do this by learning to predict relative discriminative value (quality rank) of a filter from its intrinsic properties, and by

combining the ranking scores with a diversity-inducing penalty on inter-filter similarity. Fig. 3.1 shows an overview of our approach.

The components of this automatic selection mechanism, once learned on a set of categories, can be applied to a novel target category. In that sense, it is a category-independent method for part selection. Of course, some information about the target category enters the process in the form of candidate parts, and our method can not "hallucinate" them from scratch; but it can rank them, as we show in our experiments, as accurately as a direct evaluation on thousands of examples for the category.

As its main contribution, this paper offers a practical way to speed up training detection architectures based on poselets, and exemplar SVMs, by an order of magnitude, with no loss, and in fact sometimes a moderate gain, in detection performance. This eliminates a significant computational bottleneck, as computer vision advances towards the goal of detecting thousands of categories [16]. As an additional contribution, our ranking-with-diversity approach may provide insight into what makes a good filter for object detection, with implications for design of part-based models and in descriptors and interest operators.

3.1.1 Related work

The most relevant body of work that uses part generation and selection for building detectors is the poselet model [8, 9, 17] which forms the basis for our work and which we review in detail in the next section. Alternative methods for generating part libraries/ensembles include exemplar SVMs [10], where every positive example leads to

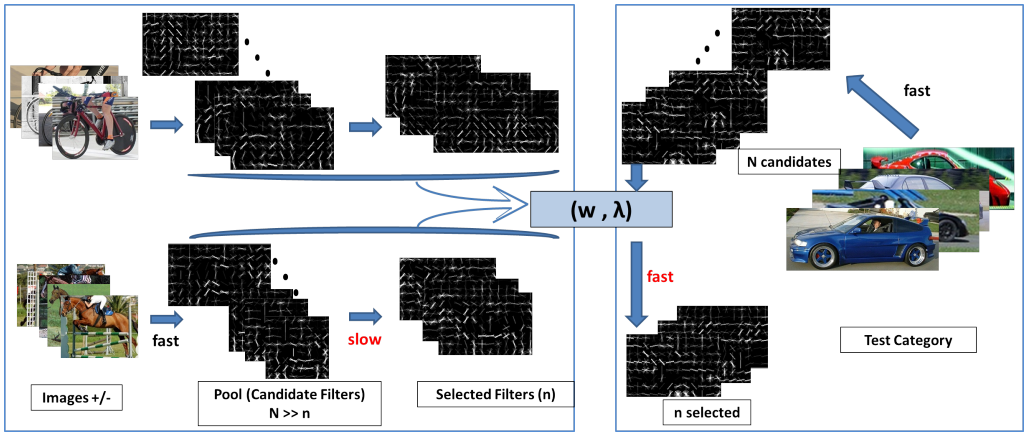

Figure 3.1: Outline of our approach: Left block shows the training pipeline which is used to obtain a linear ranker (w) and diversity tradeoff parameter λ (described in Sect. 3.3) on a set of categories. Our system improves the bottleneck of the selection procedure by learning to predict the utility of filters for a new category.

a detector (typically for an entire object). The resulting ensemble is very redundant, and may contain many poor exemplars; the hope is that these are suppressed when votes are pooled across the ensemble at detection time. In many methods detection is based on Hough-type voting by part detectors, with a library of parts built exhaustively [18], randomly [19], by a sampling mechanism [11, 20] or based on clustering [21–23]. The latter construction ensures diversity, while the former does not. Our proposal could affect all of these methods, e.g., by providing a rejection mechanism for hypothesized parts with low estimated ranking score.

Finally, a family of models in which parts are learned jointly as part of a generative model, most notably the deformable part model of [15]. Our work could be used to provide a prior on parts in this framework, as constraint in addition to deformation cost already in the model.

There has been relatively little work on predictive measures for part or filter quality. Most notably, in [13] and [24] a structured prior for HOG filters is intended to capture

spatial structure typical of discriminative image regions. [13] is the work closest to ours in spirit, and we evaluate it in our experiments. Our results show that while this "structured norm" approach is helpful, additional features that we introduce further improve our ability to distinguish good filters from bad ones.

3.2 Background

We are interested in a sliding window approach to detection [9, 12] in which an object template is specified by a filter $\mathbf{f}$. An image subwindow is represented by its feature vector $\mathbf{x}$ and is scored by the inner product $\mathbf{f}^T\mathbf{x}$. Feature vector $\mathbf{x}$ is computed by spatially dividing the subwindow to $m \times n$ cells and computing a histogram of oriented gradients for each cell. Feature vector consists of cell-level features, $\mathbf{x} = [\mathbf{x_1}; \mathbf{x_2}; \ldots; \mathbf{x_{mn}}] \in \mathbb{R}^{mnd}$, where $\forall c \in \{1, \ldots, mn\}, \mathbf{x}_c \in \mathbb{R}^d$ and d is the dimension of the cell-level features. In the same way model parameter can be broken down into $\mathbf{f} = [f_1; f_2; \ldots; f_{mn}] \in \mathbb{R}^{mnd}$. The template $\mathbf{f}$ is learned from labeled training set $\mathcal{X}, \mathcal{Y} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ by training a linear classifier, for instance by an SVM (we refer to such filters as SVM filters) or by a linear discriminant analysis (LDA filters).

We consider the category-level transfer settings: having learned filters for (training) categories $g = 1, \ldots, G$ we want to predict filter quality for a new (test) category $G + 1$. Our pipeline is outlined in Fig. 3.1. For each training category g, we start by constructing a pool of N candidate filters $\{\mathbf{f}_{g,i}\}$. Then we train a model which includes only $n \ll N$ parts. Once the models are fully trained, we can in hindsight look at the initial set of N parts and the selected set of n for each category. We train a ranking function with the

objective to reproduce the order of filter quality. Furthermore, we tune a weight which controls tradeoff between estimated rank of a filter and the diversity it adds; we want to discourage adding a part similar to the ones already selected, even if this part is highly ranked. The objective in tuning the diversity tradeoff is to as closely as possible reproduce the selection of n out of N filters done by the expensive full process.

For the test category, we construct the pool of N candidate filters $\{\mathbf{f}_{g+1,i}\}$ in the same process as for training categories. This stage is typically inexpensive, especially using the LDA method (Sect. 3.3.4). Then, we apply the learned ranker function to order the candidate parts according to their estimated relative quality. Finally, we combine suitably normalized relative scores with a diversity term tuned on training categories, and select a set of n estimated high quality candidates by a greedy procedure. This small set is used to train the full model. Thus, the expensive stage only includes n parts, instead of N. In our experiments these steps are done as part of training a poselet model [8, 9], or exemplar SVMs [10]. We briefly describe the two models below.

3.2.1 An overview of poselets for object detection

Poselets [8, 9] are semantically aligned discriminative patterns that capture parts of objects at a fixed pose and viewpoint. For person these include frontal faces, upper bodies, or side facing pedestrians; for bicycles these include side views of front wheels, etc. These patterns are *discovered* from the data using a combination of supervision in the form of landmark annotations, discriminative filter training, and a selection procedure that selects a subset of these patterns.

In more detail, each poselet is trained to detect a stable and repeatable configuration

of a subset of landmarks ("part") using HOG features and linear SVMs. This step is identical to pipelines typically used for training object detectors such as [12]. Technically, a poselet filter is obtained by randomly sampling a *seed* window covering a subset of landmarks in a positive example, then a list of matching windows, sorted by the alignment error of the landmarks (up to a similarity transform to the landmarks within the seed) is obtained. Top examples on the list (3% in our implementation) along with some negative examples are used to retrain the HOG filter, which is retained as a poselet detector. Fig. 3.2 shows examples of HOG filters, along with visualization of the average of the top 10 matching examples used to train them.

Some of the resulting poselet detectors may not be discriminative. For instance, limb detectors are often confused by parallel lines. Some others, e.g., detectors of faces and upper bodies, are more discriminative. In order to identify the set of discriminative poselets, they are evaluated as *part detectors* on the entire training set, and a subset is selected using a 'greedy coverage algorithm' that iteratively picks poselets that offer highest increase in detection accuracy at a fixed false positive rate. We can compute the detection average precision (AP) of each poselet independently, by looking at overlap between predicted and true (if any) bounding box for the part. This is what we will learn to predict in our using a discriminative ranker (Sect. 3.3).

Our poselet training and testing baseline. We use an in-house implementation of poselets training that leads to results comparable to those reported elsewhere. To isolate the effect of poselet selection, we use a simplified model that avoids some of the post-processing steps, such as, learned weights for poselets (we use uniform weights), and

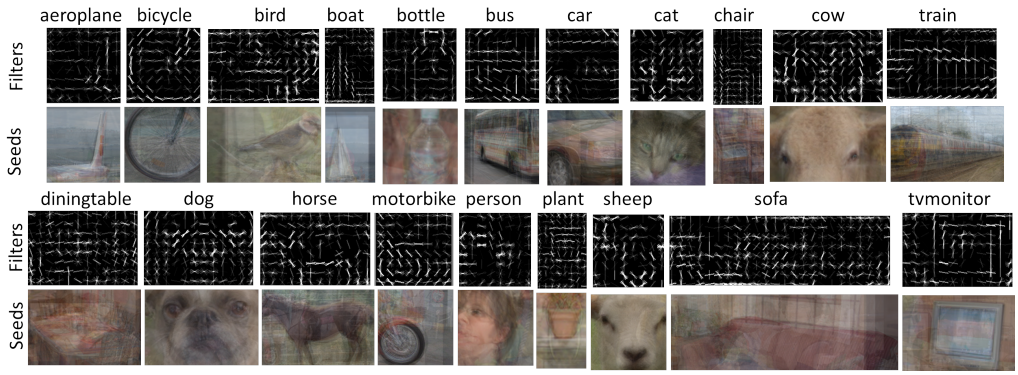

Figure 3.2: Poselet filters and the average of 10 nearest examples to its seed for various categories.

higher-order Q-poselet models (we use q-poselets, i.e., raw detection score). During training we learn 800 poselets for each category and evaluate the detector by selecting 100 poselets. Our models achieve a mean AP (MAP) of 29.0% across 20 categories of the PASCAL VOC 2007 *test* set. This is consistent with the full-blown model that achieves 32.5% MAP. The combination of Q-poselets, and learned weights per poselet, typically lead to a gain of 3% across categories. Our baseline implementation is quite competitive to existing models that use HOG features and linear SVMs, such as, the Dalal & Triggs detector (9.7%), exemplar SVMs (22.7%) [10] and DPM (33.7%). These scores are without inter-object context re-scoring, or any other post-processing.

Breakdown of the training time. Our implementation takes 20 hours to train a single model on a 6-core modern machine. About 24% of the time is spent in the initial poselet training, i.e., linear classifiers for each detector. The rest 76% of the time is spent on poselet selection, an overwhelming majority of which is spent on evaluating the 800 poselets on the training data. The actual selection, calibration and construction of the models takes less than 0.05% of the time.

3.2.2 An overview of exemplar SVMs for object detection

Exemplar SVMs [10] is a method for category representation where each positive example of a category is used to learn a HOG filter that discriminates the positive example from background instances. Thus, the number of exemplar SVMs for a given category is equal to the number of positive examples in the category, similar in the spirit to a nearest neighbor classifier. At test time each of these SVMs are run as detectors, i.e., using a multi-scale scanning window method, and the activations are collected. Overall detections are obtained by pooling spatially consistent set of activations from multiple exemplars within an image.

By design, the exemplars are likely to be highly redundant since several examples within a category are likely to be very similar to one another. Hence, a good model may be obtained by considering only a subset of the exemplars. Experimentally, we found that using only 100 best exemplars (based on the learned weights of the full model), a small fraction of the total, we obtain a performance of $\texttt{MAP} = 21.89\%$, compared to $\texttt{MAP} = 22.65\%$. We use publicly available models [1] for our experiments, and report results using **E-SVM + Co-occ** method reported in [10].

The training time scales linearly with the number of exemplars in the model. Hence, we would save significantly in training time we could quickly select a small set of relevant exemplars. We describe the details of the experimental setup in Sect. 3.5.

[1] https://github.com/quantombone/exemplarsvm

3.3 Ranking and diversity

One could attempt to predict the AP value of a filter, or some other direct measure of filter's quality, directly in a regression settings. However this is unlikely to work[2] due to a number of factors: noisy estimates of AP on training filters/categories, systemic differences across categories (some are harder than others, and thus have consistently lower performing parts), etc.

3.3.1 Learning to rank parts

Our approach instead is to train a scoring function. Given a feature representation of a part, this function produces a value (score) taken to represent the quality of the filter. Ordering a set of filters by their scores determines the predicted ranking of their quality; note that the scores themselves are not important, only their relative values are.

Let $\phi(\mathbf{f})$ be a representation of a filter $\mathbf{f}$ in terms of its *intrinsic features*; we describe the choice of ϕ in Sect. 3.3.3. We model the ranking score of $\mathbf{f}$ by a linear function $\langle \mathbf{w}, \phi(\mathbf{f}) \rangle$. The training data consists of a set of filters $\{\mathbf{f}_{g,i}\}$ for $g = 1, \ldots, G$ (training categories) and $i = 1, \ldots, N$, where N is the number of filters per category (assumed for simplicity of notation to be the same for all categories). For each $\mathbf{f}_{g,i}$ we have the estimated quality $y_{g,i}$ measured by the explicit (expensive) procedure on the training data of the respective categories. Let $\mathbf{f}_{g,i}$ be ordered by descending values of $y_{g,i}$. For $i > j$, we denote $\Delta_{g,i,j} \doteq y_{g,i} - y_{g,j}$; this measures how much better $\mathbf{f}_{g,i}$ is than $\mathbf{f}_{g,j}$.

[2] and indeed did poorly in our early experiments

We train the ranking parameters $\mathbf{w}$ to minimize the large margin ranking objective

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{g=1}^{G} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[1 - \left\langle \mathbf{w}, \delta\boldsymbol{\phi}_{g,i,j} \right\rangle\right]_+ \Delta_{g,i,j} \tag{3.1}$$

where $\delta\boldsymbol{\phi}_{g,i,j} \doteq \boldsymbol{\phi}(\mathbf{f}_{g,i}) - \boldsymbol{\phi}(\mathbf{f}_{g,j})$. and $[\cdot]_+$ is the hinge at 0. The value C determines the tradeoff between regularization penalty on $\mathbf{w}$ and the empirical ranking hinge loss. Additionally, per-example scaling by $\Delta_{g,i,j}$ is applied only to pairs on which the ranking makes mistakes; this is known as slack rescaled[3] hinge loss [25]. We minimize (3.1) in the primal, using conjugate gradient descent [27].

3.3.2 Selecting a diverse set of parts

A set of parts that are good for detection should be individually good and complementary. We can cast this as a maximization problem. Let $x_i \in \{0, 1\}$, $i \in \{1, \ldots, N\}$, denote the indicator variable that part i is selected. Let $\widehat{y}_i$ denote the (estimated) score of part i, and A_{ij} denote the similarity between parts i, j; we defer the details of evaluating A_{ij} until later. Then the problem of selecting n parts can be cast as:

$$\max_{\mathbf{x} \in \{0,1\}^N, \sum_i x_i = n} \sum_i \widehat{y}_i x_i \; - \; \lambda \sum_i \max_{j \neq i} A_{ij} x_i x_j. \tag{3.2}$$

This is a submodular function, which can be made monotone by additive shift in the values of $\widehat{y}$. For such functions, although exact maximization of this function subject to cardinaty constraint $\sum x_i = n$ is intractable, the simple greedy agorithm described below

[3] In our experiments slack rescaling performed better than margin rescaling, consistent with results reported elsewhere [25, 26].

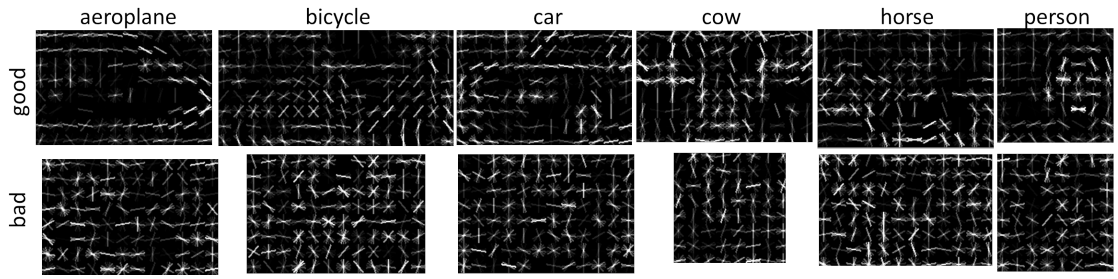

Figure 3.3: Examples of good and bad filters from the poselets model. Good filters exhibit less clutter, and stronger correlations among nearby spatial locations, than bad ones.

is known [28] to provide near-optimal solution, and to work well in practice.

First part selected is $\mathrm{argmax}_i \, \widehat{y}_i$. Now, suppose we have selected t parts, without loss of generality let those be $1, \ldots, t$. Then, we select the next part as

$$\underset{i}{\mathrm{argmax}} \left\{ \widehat{y}_i \; - \lambda \max_{j=1,\ldots,t} A_{i,j} \right\}.$$

We can further relax the diversity term, by replacing the $\max$ with the k-th order value of similarity between candidate part and those already selected. For instance, if $k = 10$, we select the first ten parts based on scores $\widehat{y}$ only, and then start penalizing candidates by the tenth highest value of similarity to selected parts. Suppose this value is σ; this means that ten parts already selected are similar to the candidate by *at least* σ. This makes it less likely that we will reject a good part because a single other part we selected is somewhat similar to it.

3.3.3 Features for part ranking

Recall that filter $\mathbf{f}$ is considered to be good if during prediction it does not confuse between a negative sub-window and a sub-window belonging to the object class. Or in

other words it results in high average precision for that object/part/poselet. Fig. 3.3 shows some examples of good and bad filters. We propose to capture the properties of a good filter by considering various low level features that can be computed from the filter itself which are described below.

- *Norm:* The first feature we consider is the ℓ_2-norm of the filter $\sqrt{\mathbf{f}^T\mathbf{f}}$. Intuitively, high norm of filter weights is consistent with high degree of alignment of positive windows similar to the seed that initiated the part, and may indicate a good part.

- *Normalized norm:* The norm is not invariant to the filter dimension ($m \times n$), which may vary across filters. Therefore we introduce *normalized norm* $\sqrt{\mathbf{f}^T\mathbf{f}}/(mn)$.

- *Cell covariance:* For good filters, the activations of different gradient orientation bins within a cell are highly structured. Neighboring gradient orientation bins are active simultaneously and majority of them are entirely suppressed. This is because the template has to account for small variations in local gradient directions in order to be robust, and if a certain gradient orientation is encouraged, its orthogonal counterpart is often penalized. For each filter $\mathbf{f} \in \mathbb{R}^{mnd}$, a $d \times d$ feature vector is obtained which captures average covariance of the filter weights within a cell.

- *Cell cross-covariance:* Similarly, there is also a strong correlation between filter weights in nearby spatial locations. Dominant orientations of neighboring cells tend to coincide to form lines, curves, corners, and parallel structures. This could be attributed to the fact that the template has to be robust to small spatial variations in alignment of training samples, and that contours of objects often exhibit such

traits. We model 4 types of features: cross-covariance between pairs of cells that are (a) horizontal (b) vertical, (c) diagonal 1 ($+45°$), and (d) diagonal 2 ($-45°$). This leads to a $4d \times d$ dimensional feature vector.

Our covariance features are inspired by [13] who used them in a generative model of filters that served as a prior for learning filters from few examples. In contrast, we use these features in a discriminative framework for selecting good filters. Our experiments suggests that the discriminative ranker outperforms the generative model (Sect. 3.4).

3.3.4 The LDA acceleration

Instead of ranking SVM filters, one can also learn to rank the filters that are obtained using linear discriminant analysis (LDA) instead [29]. The key advantage is that this can be computed efficiently in closed form as $\Sigma^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$, where Σ is a covariance matrix of HOG features computed on a large set of images, and $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ are the mean positive and negative features respectively. The parameters Σ and $\boldsymbol{\mu}_-$ need to be estimated once for all classes. *In our experiments the LDA filter by itself did not perform very well.* The LDA based detector with poselets was 10% worse in AP on bicycles, but we found that the performance of the LDA filters and that of the SVM filters are highly correlated. If the selection is effective using the LDA filters we can train the expensive SVM filters only for the selected poselets, providing a further acceleration in training time. We consider additional baseline where the ranker is trained on the LDA filters instead of the SVM filters in our experiments.

3.4 Experiments with Poselets

We perform our poselet selection on the models described in Sect. 3.2.1. For each category we have a set of 800 poselets, each with learned HOG filter trained with a SVM classifier, and its detection AP computed on the training set. We evaluate our selection in a *leave-one-out* manner – for a given category the goal is to select a subset (say of size 100) out of all the poselets by training a ranker on the remaining categories. The code can be downloaded from our project page [4].

We compare the various selection algorithms in two different settings. The first is **ranking task** where algorithms are evaluated by comparing the *predicted ranking* of poselets to the *true ranking* according to their AP. We report overlaps at different depths of the lists to measure the quality. In addition, we also evaluate the selected poselets in the **detection task**, by constructing a detector out of the selected poselets and evaluating it on the PASCAL VOC 2007 *test* set. All the poselets are trained on the PASCAL VOC 2010 *trainval* set for which the keypoint annotations are publicly available, and the images in our training set are disjoint from the test set.

3.4.1 Training the ranking algorithm

As described in Sect. 3.3.1 for each category we train a ranking algorithm that learns to order the poselets from the remaining 19 categories according to their detection AP. We normalize the APs of each class by dividing by maximum for that category to make them comparable across categories. Note that this does not change the relative ordering of

[4] http://www.umiacs.umd.edu/~ejaz/goodParts/

poselets within a class.

The learning is done according to 3.1. From the pool of all the poselet filters (19×800) we generate ordering constraints for pairs of poselets i, j for which $\Delta_{c,i,j} > 0.05$; this significantly reduces computation with negligible effect on the objective.[5] The cost of reversing the constraint is set proportional to the difference of the APs of the pair under consideration. The constant of proportionality C in Eqn. 3.1 is set using cross-validation. We consider values ranging from 10^{-13} to 10^{3}. As a criteria for cross-validation we check for ranking on the held-out set. We consider the ranked list at depth one fourth of the number of samples in the held-out set. The cross-validation score is computed as follows, $\frac{list_{predicted} \cap list_{actual}}{list_{predicted} \cup list_{actual}}$, and set using 3 fold cross validation. Note, that at any stage of the learning, the filters for the target class are not used.

3.4.2 Training the diversity model

The actual set of filters selected by the poselet model is not simply the top performing poselets, instead they are selected greedily based on highest incremental gain in detection AP. We can model this effect by encouraging diversity among the selected poselets as described in Sect 3.3.2. To do so, we first need a model of similarity between poselets. In our experiments we use a simple notion of similarity that is based on the overlap of their training examples. Note that poselets use keypoint annotations to find similar examples and provide an ordering of the training instances. For two different poselets i, j we compute the overlap of the top $r = 3\%$ (which is used for training the filters) of the ordered list of training examples Top_i and Top_j to compute the similarity,

[5] The results are not sensitive to the choice of threshold on Δ

i.e., $A_{ij} = \frac{Top_i \cap Top_j}{Top_i \cup Top_j}$. We ignore the actual filter location and simply consider the overlap between indices of training examples used. More sophisticated, but slower, versions of similarity may include computing the responses of a filter on the training examples of another.

The only parameter that remains is the term λ (Eqn. 3.3.2) controlling the tradeoff between diversity and estimated AP rank. We tune it by cross-validation. Note that unlike the previous setting for ranking where we learn to match the AP scores, here we train the diversity parameter λ to match the set of "poselets" that were *actually picked* by the poselet training algorithm. This process closely approximates the true diversity based selection algorithm. For each category, we pick a λ that matches the predicted list of other categories best on average. In practice, we found λ to be very similar across categories.

3.4.3 Selection methods considered

Below are the methods we consider for various ranking and detection tasks in the poselets framework:

- `Oracle` - poselets ordered using the poselet selection algorithm (Sect 3.2.1).
- `10%` - only 10% of the training images are used for poselet selection.
- `Random` - select a random subset of poselets.
- `Norm(svm)` - poselets ordered in descending order of ℓ_2-norm of their SVM filter.
- Σ-`Norm(svm)` - poselets ordered in descending order on SVM filters, according to $\mathbf{f}^T(\mathbf{I} - \lambda_s \mathbf{\Sigma}_s)\mathbf{f}$, where λ_s is set such that the largest eigenvalue of $\lambda_s \mathbf{\Sigma}_s = 0.9$ as defined in [13]. We construct $\mathbf{\Sigma}_s$ from top 30 filters (according to AP) from each

category to create a model of a good filter. While constructing $\mathbf{\Sigma}_s$ for one category we consider all the other category's filters.

- `Rank (svm)` - poselets ordered according to the score of the ranker trained on the SVM filters (Sect. 3.4.1).
- `Rank (lda)` - poselets ordered according to the score of the ranker trained on LDA filters (Sect 3.3.4).

In addition we consider variants with diversity term added (Sect. 3.4.2), which is shown as $+$ `Div` appended to the end of the method name.

3.4.4 Ranking results

Tab. 3.1 displays the performance of various ranking methods on the ranking task. Ranked list was looked at various depths (top $50, 100$ etc.) and its overlap was found with top 100 poselets in the groundtruth ranking (i.e. ranking according to actual AP, Sec. 3.2). Table shows number of poselets in top 100 groundtruth by considering various depths in the ranked list, averaged across categories. Note that `Rank (svm)` performs best at all the depths considered, and is closely matched by ranking using the LDA filter `Rank (lda)`. It is worth noting that the ranking task is a proxy for the real task (detection). In the next section we examine how the differences (some of them minor) between methods in Table 3.1 translate to difference in detection accuracy.

Methods	50	100	150	200
Norm(svm)	30.25	52.80	68.60	80.00
Σ-Norm(svm)	29.30	52.20	67.85	79.70
Rank(lda)	31.50	54.30	70.20	80.20
Rank(svm)	**31.55**	**55.35**	**71.20**	**81.10**

Table 3.1: The number of common filters in the ranked list for various methods and the ground truth list based on the poselet detection AP for different lengths of the list.

3.4.5 PASCAL VOC detection results

Tab. 3.2 summarizes the accuracy of the detectors, reported as the mean average precision (MAP) across the 20 categories using the model constructed from the top 100 poselets using various algorithms. We also report the speedups and relative MAP (δMAP $=$ MAP $-$ MAP$_{\texttt{oracle}}$), that various methods can provide over the actual implementation for training model consisting of 100 poselets from a pool of 800 poselets.

Ranking with SVM filters. The Random baseline performs poorly with $\delta\texttt{MAP} = -2.37\%$. Norm based ordering does well – the ℓ_2-norm based ordering already comes close with a $\delta\texttt{MAP} = -1.65\%$, while the structured norm, Σ-Norm(svm) is slightly better with a $\delta\texttt{MAP} = -1.50\%$. Our learned ranker outperforms the norm based methods (not surprising since the features include norm and the co-variance structure of the HOG cells). The ranker trained on the SVM filters achieves a $\delta\texttt{MAP} = -1.22\%$.

Adding diversity term leads to improvements across the board. Notably, the performance of the Rank(svm) + Div is indistinguishable from the original model $\delta\texttt{MAP} = +0.01\%$. Examination of the sets of 100 filters obtained with and without diversity with Rank(svm) reveals that on average (across categories) 40% of the filters are different.

All these methods provide a speedup of $8\times$ in the poselet selection step relative to `Oracle`, and an overall speed up of $3\times$, since the initial training of expensive SVM filters still has to be done which consumes 24% of the overall training time as described in Sect. 3.2.1 (except for `Random` which provides a speedup of $8\times$, but at significant loss of accuracy). Finally, an alternative way to achieve such speedup is to evaluate the AP of the filters directly, but on only the fraction of the data; this `10%` method does significantly worse than our proposed methods, and provides a smaller speedup of $2.4\times$ since all the filters need to be evaluated on `10%` of the data. One likely reason for the low performance: most poselets, including useful ones, are rare (hence the pretty low APs even for the top performing parts), and subsampling the training set might remove almost all true positive examples for many parts, skewing the estimated APs. Larger subsets, e.g., 25% would lead to even smaller speedups, $1.9\times$ in this case.

Ranking with LDA filters. Next we consider LDA filters, and we find the the performance of the selection of poselets based on the LDA filters is slightly worse. The diversity based ranker trained on the LDA filters, `Rank(lda)+Div`, achieves a $\delta\texttt{MAP} = -0.84\%$. The key advantage of ranking using the LDA filters is that it speeds up the initial poselet training time as well, since only 100 poselets are further trained using SVM bootstrapping and data-mining. Thus the overall speed up provided by this procedure is $8\times$, almost an order of magnitude. On a six-core machines it takes about 2.5 hours to train a single model, compared to 20 hours for the original model. Note that we only use the LDA filter for ranking as we found that the LDA filters themselves are rather poor for detection on a number of categories. Notably, the bicycle detector was 10% worse – the LDA

Method	VOC 2007 test MAP	δMAP	Training speedup Initial	Selection	Overall
Oracle	29.03				
Random	26.66	-2.37	$8\times$	$8\times$	$8\times$
10%	27.78	-1.25	$1\times$	$4.4\times$	$2.4\times$
Norm(svm)	27.38	-1.65	$1\times$	$8\times$	$3\times$
Norm(svm) + Div	28.34	-0.69	$1\times$	$8\times$	$3\times$
Σ-Norm(svm)	27.53	-1.50	$1\times$	$8\times$	$3\times$
Σ-Norm(svm) + Div	28.51	-0.52	$1\times$	$8\times$	$3\times$
Rank(svm)	27.81	-1.22	$1\times$	$8\times$	$3\times$
Rank(svm) + Div	29.04	$+0.01$	$1\times$	$8\times$	$3\times$
Rank(lda) + Div	28.19	-0.84	$8\times$	$8\times$	$8\times$
Rank(lda) + Div ($2\times$ seeds)	**29.46**	**+0.43**	$8\times$	$8\times$	$8\times$

Table 3.2: Performance of poselet selection algorithms on PASCAL VOC 2007 detection. Table 3.3 shows per category results for various methods using the poselet framework.

based wheel detector has many false positives on wheels of cars, which the hard-negative mining stage of SVM training learns to discriminate.

The $2\times$ poselets experiment. We can select twice as many seeds and select an even better set of 100 poselets using the diversity based ranker based on the LDA filters. This has a negligible effect on the training time as the seed generation and LDA filter computation takes a small amount of additional time ($< 1\%$). However, this improves the performance which is better than the original model with δMAP $=$ **0.43**%, while still being an *order of magnitude* faster than the original algorithm.

PASCAL VOC 2010 results. We evaluated the oracle and the best performing method (Rank(lda) + Div (2x seeds)), on the PASCAL VOC 2010 detection test set and achieved a δMAP $=$ **0.56**%.

	Oracle	Rand	10%	Norm (Φ_{SVM})	Norm + Div (Φ_{SVM})	Σ-Norm (Φ_{SVM})	Σ-Norm + Div (Φ_{SVM})	Rank (Φ_{SVM})	Rank + Div (Φ_{SVM})	Rank + Div (Φ_{LDA})	Rank (Φ_{LDA}) +Div (2× seeds)
aeroplane	**32.37**	30.37	31.10	29.08	29.39	27.83	27.34	25.62	26.20	28.30	31.29
bicycle	50.00	47.75	49.81	43.46	47.08	44.69	48.49	43.76	**53.57**	50.79	47.49
bird	12.82	10.51	11.24	**13.54**	13.34	13.42	13.22	11.74	12.02	11.38	10.74
boat	16.36	14.30	14.30	**17.90**	17.47	17.61	16.41	17.76	17.32	16.70	15.17
bottle	31.57	29.85	26.76	32.37	33.06	32.94	**33.16**	32.15	31.14	29.72	31.43
bus	41.30	41.98	40.98	37.61	37.62	37.05	37.74	**42.59**	39.00	40.06	41.80
car	56.00	52.48	53.24	54.05	55.66	52.84	**56.50**	52.62	54.75	54.97	55.46
cat	**20.84**	16.37	19.59	20.52	20.62	20.52	20.05	20.52	19.43	19.80	20.31
chair	19.20	15.63	17.70	18.76	**19.34**	18.84	18.64	17.45	17.27	16.90	17.48
cow	37.55	32.73	37.68	32.31	35.43	32.97	34.14	38.32	37.22	35.59	**39.13**
diningtable	14.51	11.71	14.87	14.63	16.41	14.73	15.94	14.56	**18.11**	14.69	15.41
dog	**17.04**	11.19	15.15	14.12	15.35	14.94	15.57	15.14	15.27	14.91	16.72
horse	37.63	33.01	36.38	37.37	41.25	38.84	42.47	41.91	40.28	36.73	**43.03**
motorbike	35.91	34.30	36.86	31.07	31.47	30.93	33.87	33.82	37.50	38.64	**39.27**
person	**36.65**	30.75	34.81	34.43	33.95	34.59	34.50	35.23	35.96	33.06	34.82
pottedplant	13.14	11.79	11.61	12.25	12.78	12.25	12.84	12.72	13.06	13.21	**13.90**
sheep	31.87	29.56	26.81	30.10	31.43	31.01	31.88	28.52	**33.51**	31.14	31.89
sofa	23.35	23.46	25.80	22.08	21.99	22.08	22.53	20.92	23.50	23.12	**28.77**
train	24.31	**25.60**	22.52	19.84	21.18	19.91	21.75	20.30	25.29	25.42	24.94
tvmonitor	28.21	29.77	28.30	32.03	32.03	32.67	**33.16**	30.50	30.34	28.72	30.06
mean AP	29.03	26.66	27.78	27.38	28.34	27.53	28.51	27.81	29.04	28.19	**29.46**

Table 3.3: Per Category Results for Poselets. Best results are highlighted in bold. Best mean average precision of 29.46 is obtained using Rank (Φ_{LDA}) + Div (2× seeds). Note that this method resulted in 8× speed up of the training pipeline.

Visualization of Selected Poselets. Figures 3.4 and 3.5 show visualization of parts selected by rankSVM and rankSVM+diversity. Figures show top 10 filters along with their seeds (positive chips).

Affect of k in max-k criteria for Diversity. We tested different values of k in the max-k criteria for diveresity selection as described in the paper. We got $\Delta(AP) = -0.75$, $\Delta(AP) = -0.50$ and $\Delta(AP) = +0.01$ for $k = 1$,$k = 5$ and $k = 10$ respectively. Here $\Delta(AP) = AP_{method} - AP_{oracle}$ and $AP_{oracle} = 29.03$.

3.5 Experiments with exemplar SVMs

Here we report experiments on training exemplar SVMs. As described in Sect. 3.2.2, exemplar SVMs' training time scales linearly with the number of positive examples in the category. On the PASCAL VOC 2007 dataset, each category has on average 630 exemplars. Our goal is to select a set of 100 exemplars such that they reproduce the perfor-

mance of the optimal set of 100 exemplars. This is obtained as follows: we use the model trained using all the exemplars and use the weights learned per exemplar in the final scoring model as an indicator of its importance. The `oracle` method picks the 100 most important exemplars, and obtains a performance of $\texttt{MAP} = 21.89\%$.

Unlike poselet filters, some of these exemplars are likely to be rare. Thus even though the filter looks good, it may not be useful for detection since it is likely to detect only a small number of positive examples. Hence, we need to consider the *frequency* of the filter, in addition to its quality as a measure of importance. We use a simple method for frequency estimation. Each exemplar filter is evaluated on the every other positive instance, and the highest response is computed among all locations that have overlap $> 50\%$. Let, s_{ij}, denote the normalized score of exemplar i on instance j, i.e, $s_{ii} = 1$. Then, the frequency of the i^{th} filter is the number of detections with score $> \theta$, where θ is set to be the 95 percentile of the entries in s. The overall quality of the filter $\mathbf{f}$ is the sum of score obtained from the ranker and is frequency, $\texttt{Rank}(\mathbf{f}) + \texttt{Freq}(\mathbf{f})$.

The same metric can be used for diversity. In our experiments we say that $\tau = 5\%$ of the nearest exemplars are considered similar. For each category the ranker itself was trained on the poselets of the other 19 categories, i.e., we use `Rank(lda)` model described in Sect. 3.4.3. The diversity tradeoff parameter λ is estimated again by cross-validation within the 19 categories.

To summarize, our overall procedure for exemplar selection is, (a) we train an LDA filter for each exemplar, (b) using the ranker (trained on poselet model for the training categories) select a set of 100 filters and associated exemplars, (c) train the full model with SVM filters for these 100 exemplars. Steps (a) and (b) are relatively inexpensive,

hence the training time is dominated by step (c). Compared to the oracle model with 100 exemplars, our fast selection procedure offers a $6.3\times$ speedup.

3.5.1 PASCAL VOC detection results

Here we compare several selection strategies listed below:

- `Oracle` - top 100 filters picked according to learned weights (as described earlier)
- `Random` - a random set of 100 filters.
- `Freq` - the set of 100 most frequent filters.
- `Rank(lda)` - the set of 100 highest ranked filters according to the LDA ranker.
- `Rank(lda)` + `Freq` - the set of 100 filters according to the rank and frequency.
- `Rank(lda)` + `Freq` + `Div` - previous step with diversity term added.

Tab. 3.4 shows the performance of various methods on the PASCAL VOC 2007 dataset reported as the mean average precision (`MAP`) across 20 categories. The `Oracle` obtains 21.89%, while `Random` does poorly at 18.53%. Frequency alone is insufficient, and does even worse at 16.23%. Similarly rank alone is insufficient with performance of 17.93%. Our ranker combined with frequency obtains 18.75%, while adding the diversity term improves the performance to 19.62%. Figure 3.6 shows top 5 selected exemplars by Rank(lda)+Freq+Div. Note that we obtain this result using the model trained on the poselet filters and using LDA for training the exemplars. Replacing this with SVM filters may close the gap even further as we observed in the poselet based experiments.

Method	MAP on VOC 2007 test
Oracle	21.89
Random	18.53
Freq	16.23
Rank(lda)	17.93
Rank(lda) + Freq	18.75
Rank(lda) + Freq + Div	19.62

Table 3.4: Performance of selection algorithms for detection on the PASCAL VOC 2007 dataset. All these methods provide a speed up of $6.3\times$ relative to the Oracle as there are on average 630 exemplars per category. Table 3.5 shows per category results for various methods using the exemplar SVM framework.

	Oracle	Rand	Freq	Rank(lda)	Rank(lda) + Freq	Rank(lda) + Freq + Div
aeroplane	**23.58**	10.63	10.44	9.66	9.91	21.83
bicycle	41.72	**42.56**	34.82	38.86	40.66	40.30
bird	9.23	9.18	**9.25**	9.14	9.12	9.13
boat	**13.56**	10.53	9.49	10.18	9.36	10.04
bottle	**11.54**	9.34	9.18	9.46	9.19	9.23
bus	**39.56**	33.05	30.87	32.77	34.84	36.38
car	**37.96**	33.92	17.34	29.89	32.39	32.41
cat	9.41	9.38	9.36	**9.53**	9.15	9.21
chair	**9.98**	9.68	9.36	9.22	9.11	9.12
cow	**19.58**	14.64	14.69	15.24	18.66	18.54
diningtable	10.07	9.47	**10.29**	9.91	9.96	9.81
dog	**10.03**	9.41	9.32	9.31	9.23	9.22
horse	**41.70**	35.93	27.38	30.17	33.32	32.34
motorbike	**33.26**	32.28	25.33	31.67	32.64	32.64
person	**15.58**	10.08	9.65	9.90	10.40	11.04
pottedplant	9.62	9.51	**9.62**	9.41	9.10	9.23
sheep	**24.33**	18.69	14.47	18.51	20.23	22.07
sofa	**13.30**	9.95	9.77	11.07	11.22	11.26
train	**33.23**	25.20	26.14	28.09	26.98	28.08
tvmonitor	30.49	27.16	27.82	26.70	29.47	**30.58**
mean AP	**21.89**	18.53	16.23	17.93	18.75	19.62

Table 3.5: Per Category Results for Exemplar SVMs. Note how Freq and Div both play an important role for esvm along with ranker (result closest to the Oracle is obtained when all three,i.e. Rank, Freq and Div are used together). All these methods result in a speed-up of $6.3\times$ relative to the Oracle.

3.5.2 An analysis of bicycle HOG filters

Finally, we look at the bicycle category to get some insight into the ranker. We take the filters obtained from the poselets model, as well as exemplar SVMs. To decouple the effect of frequency we only consider side-facing bicycle exemplars. The assumption here is that all side-facing exemplars have the same frequency.

Fig. 3.7 (top) shows a scatterplot of the score obtained by the ranker (higher is better) and the true ranks of the filters (lower is better) for poselets and exemplar SVMs. For poselets there is a strong (anti) correlation between the predicted score and quality (*correlation coefficient = -0.64*). For exemplar SVMs, the prediction is weaker, but it does exhibit high (anti) correlation (*correlation coefficient = -0.42*). Fig. 3.7 (bottom) shows the 10 least and highest ranked side-facing exemplars. The ranker picks the exemplars that have high figure-ground contrast revealing the relevant shape information and little background clutter.

3.6 Conclusion

We described an automatic mechanism for selecting a diverse set of discriminative parts. As an alternative to the expensive explicit evaluation that is often the bottleneck in many methods, such as poselets, this has the potential to dramatically alter the tradeoff between accuracy of a part based model and the cost of training. In our experiments, we show that combined with LDA-HOG, an efficient alternative to SVM, for training the part candidates, we can reduce the training time of a poselet model by an order of magnitude, while actually improving its detection accuracy. Moreover, we show that our approach

to prediction of filter quality transcends specific detection architecture: rankers trained for poselets allow efficient filter/exemplar ranking for exemplar SVMs as well. This also reduced the training time for exemplar SVMs by an order of magnitude while suffering a small loss in performance.

The impact of such a reduction would be particularly important when one wants to experiment with many variants of the algorithm – situation all too familiar to practitioners of computer vision. Our work suggests that it is possible to evaluate the discriminative quality of a set of filters based purely on their intrinsic properties. Beyond direct savings in training time for part-based models, this evaluation may lead to speeding up part-based detection methods at *test time*, when used as an attention mechanism to reduce number of convolutions and/or hashing lookups.

Our plans for future work include investigation of the role of class affinity in generalization of part quality; e.g., one might benefit from using part ranking from vehicle classes when the test class is also a vehicle.

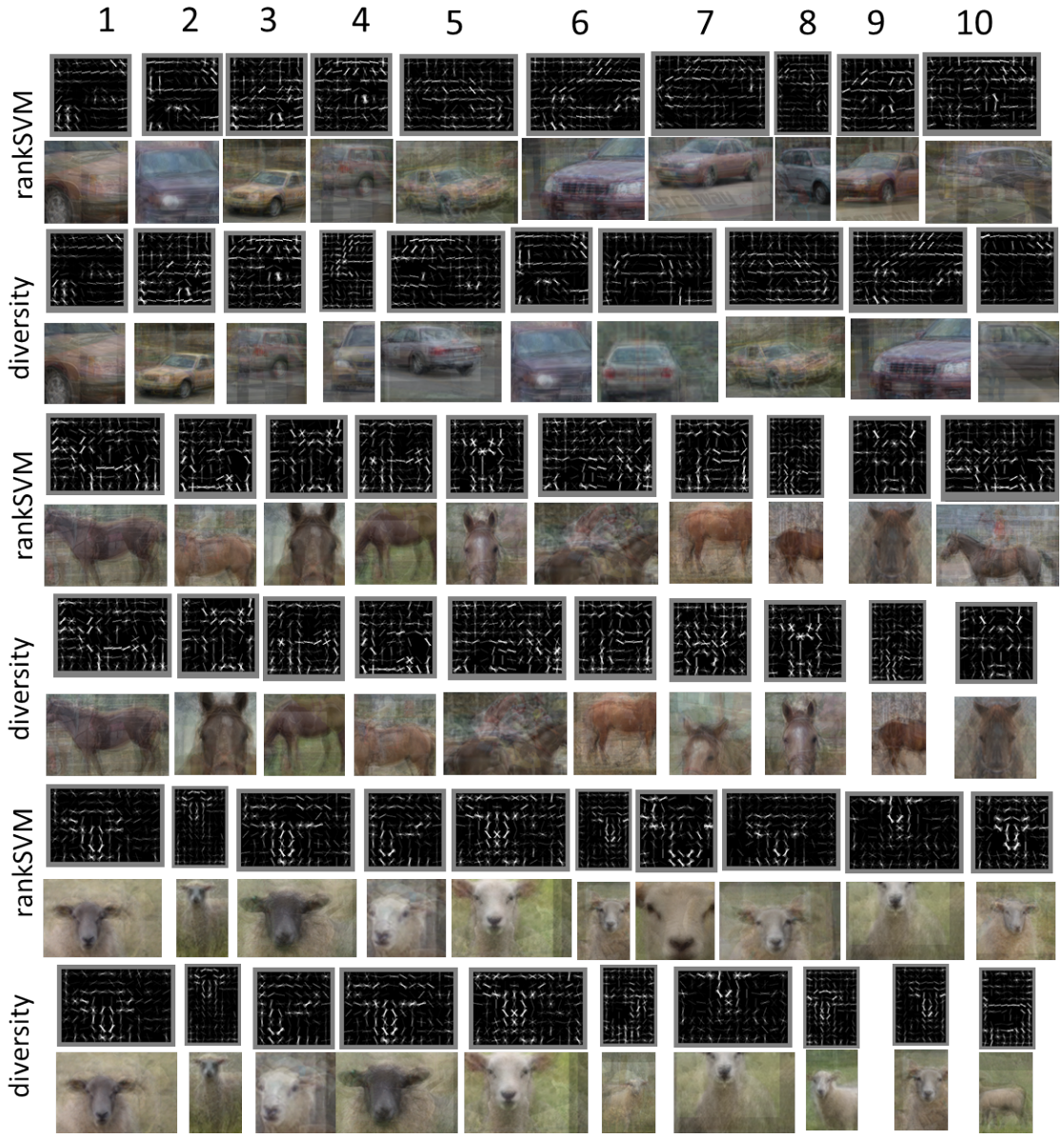

Figure 3.4: Top 10 poselets (filter and seeds) per category discovered using rankSVM (top 2 rows). Re-ordering using our diversity selection method (bottom 2 rows). Note that parts that are ranked higher by rankSVM might not end up high in the ordering by diversity. Diversity selection picks diverse set of parts for instance in case of sheep, instead of just picking faces which might have high prediction capability by themselves, it also picks side veiw of sheep (last part).

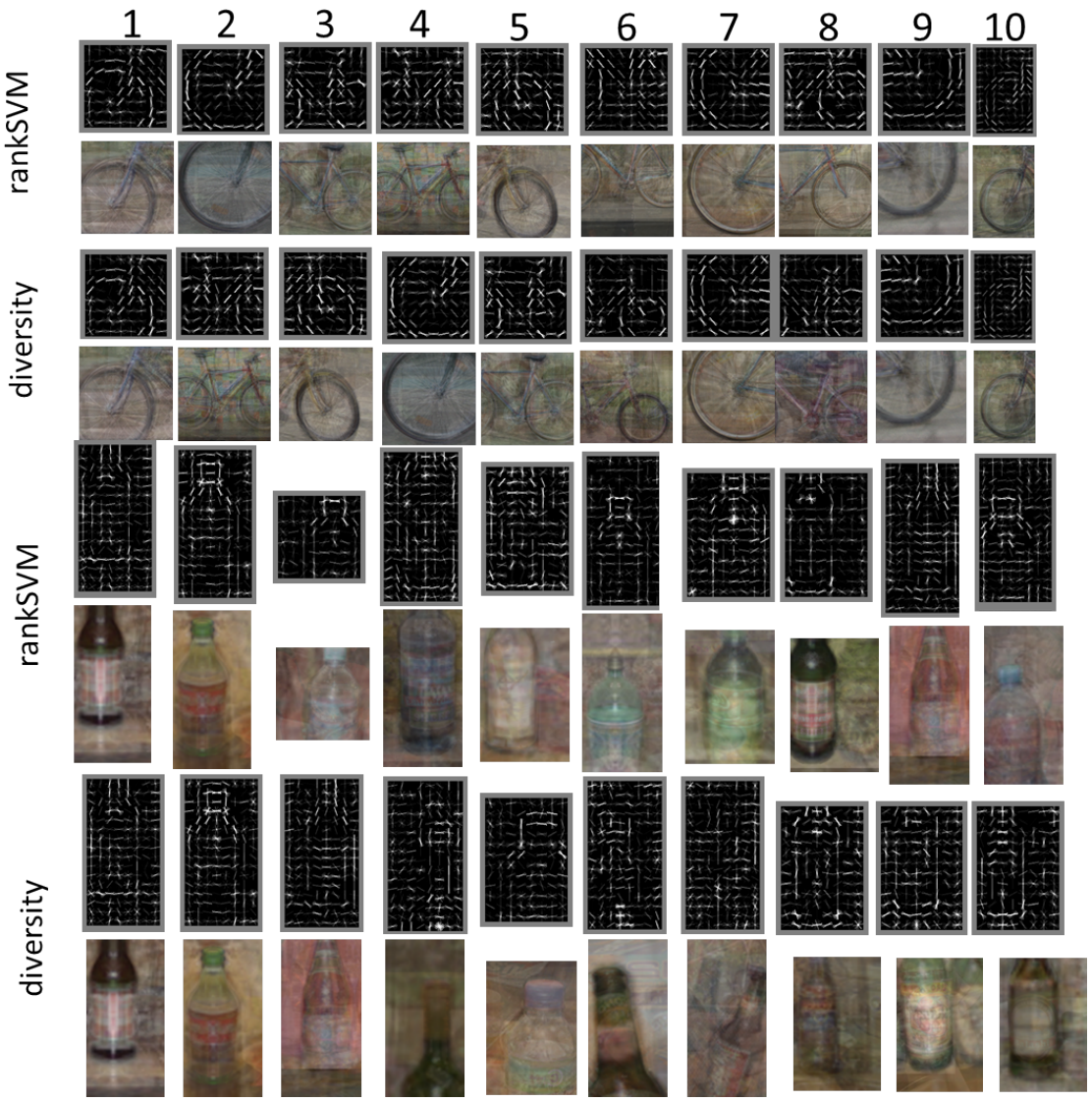

Figure 3.5: More examples of visualization of poselets selected by rankSVM and by rankSVM + diversity.

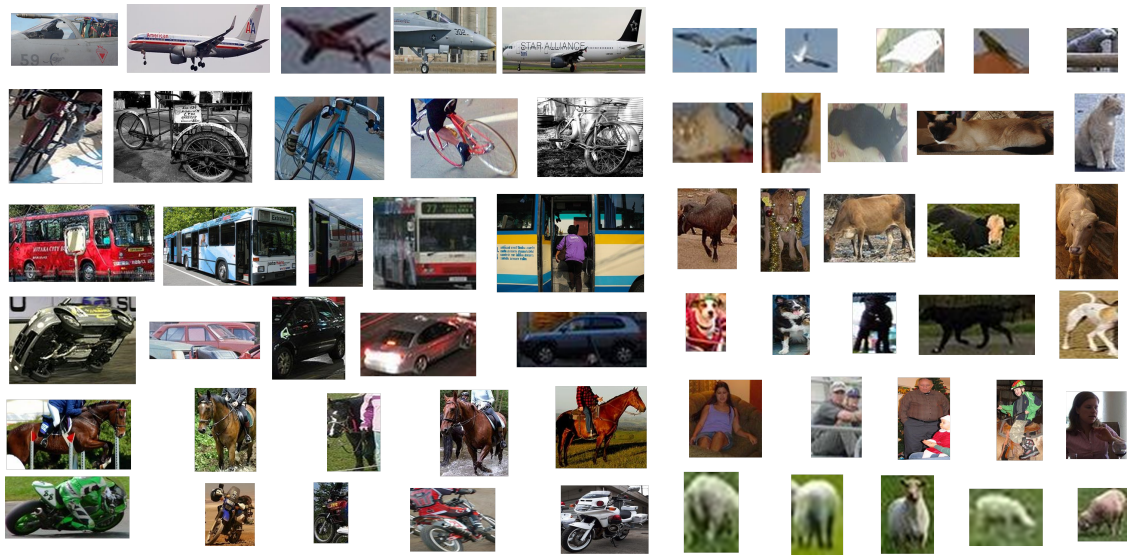

Figure 3.6: Top 5 exemplars selected by our method `Rank(lda) + Freq + Div` . Categories in row major order - aeroplane, bird, bicycle, cat, bus, cow, car, dog, horse, person, motorbike and sheep.

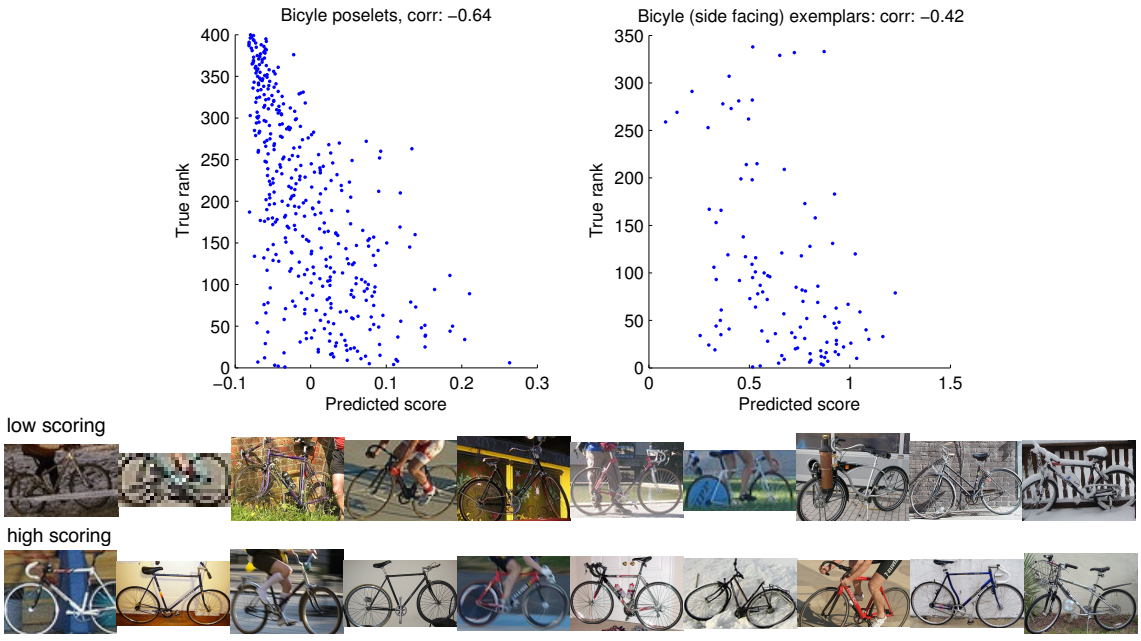

Figure 3.7: **An analysis of bicycle filters**. (Top-left) Scatter plot of true ranks and the ranker score of the bicycle poselets. (Top-right) the same for all the exemplars of side-facing bicycles. The high scoring side-facing exemplars (Bottom row) exhibit high contrast and less clutter than the low scoring exemplars (Middle row).

Chapter 4: Composite Discriminant Factor Analysis

In this chapter we present a linear dimensionality reduction method, Composite Discriminant Factor (CDF) analysis, which searches for a discriminative but compact feature subspace that can be used as input to classifiers that suffer from problems such as multi-collinearity or the curse of dimensionality. The subspace selected by CDF maximizes the performance of the entire classification pipeline, and is chosen from a set of candidate subspaces that are each discriminative. Our method is based on Partial Least Squares (PLS) analysis, and can be viewed as a generalization of the PLS1 algorithm, designed to increase discrimination in classification tasks. We demonstrate our approach on the UCF50 action recognition dataset, two object detection datasets (INRIA pedestrians and vehicles from aerial imagery), and machine learning datasets from the UCI Machine Learning repository. Experimental results show that the proposed approach improves significantly in terms of accuracy over linear SVM, and also over PLS in terms of compactness and efficiency, while maintaining or improving accuracy. This work has been published in [30].

4.1 Introduction

Dimensionality reduction methods have been popular in the computer vision community [31] as preprocessing tools that deal with the increasing dimensionality of input features. The literature includes linear methods [32–34]; non-linear methods, some of which are kernelized versions of linear methods [35–38]; and feature selection methods [31]. We focus on linear feature construction methods that obtain compact but predictive features by linear transformations, motivated by the task of object detection, which involves high-dimensional features constructed from dense feature grids (e.g., HOG [39, 40], pyramidal HOG [41], dense SIFT [42]) and a sliding window detection step that repeatedly applies classifiers to features constructed from image sub-windows at varying scales, translations, and rotations. The sliding window detection process benefits from linear projections in various ways. For instance, new samples are efficiently projected into the subspace by matrix multiplication and the high-dimensional training data does not need to be stored as it is for kernel methods, reducing memory and computational requirements. Additionally, linear projection can be performed efficiently by first extracting a feature grid for the entire image and then performing linear convolution [40], thus avoiding redundant computation of features included in multiple windows at different offsets. Consequently, many state-of-the-art approaches use linear classifiers, typically linear SVM [35], not only for detection but also for other tasks (e.g., action recogntion [43]).

Motivated by these trends, we propose a new approach, Composite Discriminative Factor (CDF) analysis, that selects one or more linear projection vectors to produce a compact and discriminative subspace, optionally followed by a non-linear classification

step (which is computationally cheap on low-dimensional inputs). This process is based on Partial Least Squares (PLS) [44, 45], a class of methods which model the relationship between two or more sets of observed variables via a set of latent variables chosen to maximize the covariance between the sets of observed variables. More specifically, our approach is based on the most frequently used variants of PLS [45], PLS1 and PLS2, both of which are used for regression by a process that iteratively obtains a projection vector that maximizes covariance between the input and response variables. Instead of using PLS directly, as has been done previously [46, 47], we use PLS internally to generate compact subspaces that improve the performance of our entire classification pipeline.

Our approach is based on the observations that 1) maximizing covariance between the input features and response variables does not necessarily yield a compact feature space for the purpose of classification, and 2) linear combinations of PLS factors obtained by performing regression from the latent space to the response variables are much more compact and almost as discriminative as the factors themselves. For binary classification, the composite is a projection vector. By varying how many factors are used to create a composite, we create a number of candidate projection vectors. Taking advantage of the PLS deflation operation, we iteratively alternate between the selection of a composite direction and deflation to obtain multiple projection vectors that define a multidimensional latent subspace. The number of composites we deflate by and the number of PLS factors per each composite parametrize a set of candidate subspaces. Using cross-validation and best-first search, we select from these subspaces the one that maximizes performance for the entire classification pipeline.

One appealing property of our approach is that the set of candidate CDF subspaces

includes the original PLS subspace, so it can be viewed as a generalization of PLS. In addition, subject to mild constraints, approaches other than PLS can be used to propose projection vectors at each iteration. We show empirically that our process not only outperforms PLS and other state-of-the-art baseline approaches on a number of datasets, but it does so with only one- or two-dimensional subspaces. We demonstrate the performance of our approach on the tasks of pedestrian detection on the INRIA Pedestrian dataset [39], vehicle detection in aerial images that we will make publicly available, and action recognition on the UCF50 [48] dataset. In addition, we demonstrate our approach on four public datasets from the UCI Machine Learning repository [49]. Our experiments suggest that many algorithms could be improved by replacing linear SVM with CDF, since linear SVM is a common component of many state-of-the-art computer vision algorithms that depend on linear projections of high-dimensional data.

4.1.1 Related work

Linear methods have been used in the field of computer vision for dimensionality reduction or directly for classification. For example, Principal Component Analysis has been used as a dimensionality reduction approach for face recognition by [50], followed by Linear Discriminant Analysis (LDA) for face [51], pedestrian, and object recognition [52]. Other methods, such as Canonical Correlation Analysis (CCA) have also been applied to vision [53].

A popular linear classifier and descriptor combination currently employed by a large number of state-of-the-art vision approaches is linear SVM [32] and Histograms of Ori-

ented Gradients (HOGs), initially applied by Dalal and Triggs [39] to detect pedestrians. Subsequently, improved human detectors have been proposed that can handle partial occlusion [54]. More general deformable part models (DPM) have been proposed that model objects as a set of part filters anchored to a root filter that are applied to modified HOG features, and trained using an extension of linear SVM, called Latent SVM. Recently, Malisiewicz et al. train linear SVM classifiers on HOG descriptors of each in a one-vs-all fashion to every positive instance (or exemplar) available in the training set [55]. Other approaches using these building blocks include: branch-and-bound detection applied to linear SVMs for efficient search [56]; coarse-to-fine object localization [41, 57]; scale invariant detection at multiple resolutions, in which small instances are detected with rigid templates and large instances are detected by deformable part models [58]; active learning [59], where a linear classifier is used to identify uncertain windows that need to be labeled manually; and pose-estimation [60] using an approach similar to DPM. Linear SVM has also been used in other state-of-the-art applications that do not rely on HOG, e.g., multiclass action recognition using ActionBank features [43], among many others.

Other linear classifier approaches have been proposed as well. In particular, Partial Least Squares (PLS) [44], has been recently applied to the problem of human and vehicle detection [46, 47], largely due to its ability to efficiently handle high dimensional data. Unlike PCA [33], PLS can be used as a class-aware dimension reduction tool, and unlike other class-aware dimension reduction tools, such as LDA [33, 34] or CCA [34], it can handle very high-dimensional data and its associated problems (multi-collinearity, in particular). While many PLS extensions exist such as Canonical PLS (CPLS) and Canonical

Power PLS (CPPLS) [61], Kernel PLS [62], and others [45], we will focus on extensions to the standard linear PLS approach with the goal of improving existing linear approaches that are used in many of the vision systems described above. Our work is motivated by our observation that PLS often outperforms linear SVM but that it also requires a larger linear subspace (linear SVM can be seen as projecting into a single-dimensional subspace).

Our contribution consists of a new approach, CDF, which is based on PLS but yields more compact linear subspaces that can be used for training classifiers. The benefit of lower dimensional subspaces, provided that they preserve discriminability, is not just computational–more complex classification approaches often generalize better if presented with samples that lie in a lower dimensional subspace. In the following sections, we will briefly summarize PLS, introduce our approach, and present experimental results on pedestrian detection, vehicle detection, action recognition, and benchmark machine learning datasets.

4.2 Partial Least Squares

A number of Partial Least Squares (PLS) variants model relations between two or more sets of observed variables through a set of latent variables; many of these are discussed in detail in [44, 45]. We briefly summarize the most frequently used variants, PLS1 and PLS2 [45], which relate two sets of observed variables $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times q}$, and are generally used for regression problems. Here, n is the number of observed samples, p is the dimensionality of samples from $\mathbf{X}$ and q is the dimensionality of samples from $\mathbf{Y}$. PLS1 is the special case where $q = 1$, while PLS2 is the more general case where

$q > 1$. PLS decomposes the zero-mean matrices $\mathbf{X}$ and $\mathbf{Y}$ as follows:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathsf{T}} + \mathbf{E}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^{\mathsf{T}} + \mathbf{F}$$

where $\mathbf{T}$ and $\mathbf{U}$ are $n \times f$ matrices containing f latent vectors $\mathbf{t_i}$ and $\mathbf{u_i}$ (the coefficients obtained by projecting into the latent space), $\mathbf{P} \in \mathbb{R}^{p \times f}$ and $\mathbf{Q} \in \mathbb{R}^{q \times f}$ contain the loadings (the basis vectors which minimize squared reconstruction error), and $\mathbf{E} \in \mathbb{R}^{n \times p}$ and $\mathbf{F} \in \mathbb{R}^{n \times q}$ are the residuals that result from using only f latent vectors to reconstruct $\mathbf{X}$ and $\mathbf{Y}$ (a low rank approximation similar to keeping only the dominant f eigenvectors for PCA). Usually the PLS decomposition is obtained by the nonlinear iterative partial least squares (NIPALS) algorithm [44], summarized in Algorithm 4.1, which iteratively constructs $\mathbf{T}$, $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{C}$ one column at a time by finding at each iteration i the weight vectors $\mathbf{w_i}$ and $\mathbf{c_i}$ that maximize the covariance between latent coefficients $\mathbf{t_i} = \mathbf{X}\mathbf{w_i}$ and $\mathbf{u_i} = \mathbf{Y}\mathbf{c_i}$:

$$[cov(\mathbf{X}\mathbf{w_i}, \mathbf{Y}\mathbf{c_i})]^2 = \max_{||\mathbf{r}||=||\mathbf{s}||=1} [cov(\mathbf{X}\mathbf{r}, \mathbf{Y}\mathbf{s})]^2.$$

The NIPALS algorithm finds the $\mathbf{w_i}$ and $\mathbf{c_i}$ that maximize the covariance from above by obtaining the leading eigenvector of $\mathbf{X}^{\mathsf{T}}\mathbf{Y}\mathbf{Y}^{\mathsf{T}}\mathbf{X}\mathbf{w_i} = \lambda\mathbf{w_i}$. The vector $\mathbf{c_i}$, which is the leading eigenvector of a related problem, can be computed from $\mathbf{w_i}$, and is also obtained by NIPALS in Algorithm 4.1 via the power iteration loop on lines 2–8. Once weight vectors $\mathbf{w}$ and $\mathbf{c}$ are obtained, the normalized score vector $\mathbf{t_i} = \mathbf{X}\mathbf{w_i}/||\mathbf{X}\mathbf{w_i}||$ is computed. The matrix $\mathbf{X}$ is deflated by its rank-one reconstruction from $\mathbf{t_i}$, and $\mathbf{Y}$ is deflated by the

Figure 4.1: PLS Algorithm (NIPALS version)

1: **for** $i = 1, \ldots, f$ **do**
2: $\quad \mathbf{u_i} \leftarrow \mathbf{y}_1 / ||\mathbf{y}_1||$
3: $\quad$ **repeat**
4: $\quad\quad \mathbf{w_i} \leftarrow \mathbf{X}^\mathsf{T}\mathbf{u_i} / ||\mathbf{X}^\mathsf{T}\mathbf{u_i}||$
5: $\quad\quad \mathbf{t_i} \leftarrow \mathbf{X}\mathbf{w_i} / ||\mathbf{X}\mathbf{w_i}||$
6: $\quad\quad \mathbf{c_i} \leftarrow \mathbf{Y}^\mathsf{T}\mathbf{t_i} / ||\mathbf{Y}^\mathsf{T}\mathbf{t_i}||$
7: $\quad\quad \mathbf{u_i} \leftarrow \mathbf{Y}\mathbf{c_i}$
8: $\quad$ **until** convergence
9: $\quad \mathbf{X} \leftarrow \mathbf{X} - \mathbf{t_i}\mathbf{t_i}^\mathsf{T}\mathbf{X}$
10: $\quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t_i}\mathbf{t_i}^\mathsf{T}\mathbf{Y}$
11: **end for**

rank-one component of the regression of $\mathbf{Y}$ on $\mathbf{t_i}$ (Alg. 4.1, lines 9–10). The deflation step guarantees that subsequent weight vectors $\mathbf{w_{i+1}}$ and resulting score vectors $\mathbf{t_{i+1}}$ explain only the residuals, and thus are independent, i.e. $\mathbf{T}^\mathsf{T}\mathbf{T} = \mathbf{I}$ and $\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}$, where $\mathbf{t_i}$ and $\mathbf{w_i}$ are the i^{th} colums of $\mathbf{T}$ and $\mathbf{W}$. It can be shown that $\mathbf{P} = \mathbf{X}^\mathsf{T}\mathbf{T}$ minimizes reconstruction error $||\mathbf{E}||_2$. Because the columns of $\mathbf{W}$ are computed from deflated data, we compute a matrix $\mathbf{W}_* = \mathbf{W}(\mathbf{P}^\mathsf{T}\mathbf{W})^{-1}$ that corrects for the deflation step so that we can obtain the latent scores (or coefficients) of $\mathbf{X}$ by a linear projection, $\mathbf{T} = \mathbf{X}\mathbf{W}_*$.

PLS classification can be performed by letting $\mathbf{X}$ be the input features and $\mathbf{Y}$ be the $n \times c$ class indicator matrix for multiclass classification or a $n \times 1$ indicator vector for the binary case. If PLS is used for feature extraction, then f factors are extracted as linear combinations of the input features, and some other classifier (e.g., QDA) is applied to the factors $\mathbf{T} = \mathbf{X}\mathbf{W}_*$. Note that because $\mathbf{T}^\mathsf{T}\mathbf{T} = \mathbf{I}$, the projected data is also *whitened* in the process, a preprocessing step that often improves classifier performance. Alternatively, classification can be performed by linear regression, predicting the indicator matrix from the input features by $\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{G}$, where $\mathbf{B} = \mathbf{W}(\mathbf{P}^\mathsf{T}\mathbf{W})^{-1}\mathbf{T}^\mathsf{T}\mathbf{Y} = \mathbf{W}_*\mathbf{T}^\mathsf{T}\mathbf{Y}$ and $\mathbf{G}$ is a residual matrix. In subsequent sections, we denote the vector $\mathbf{B}$ by

Figure 4.2: CDF Algorithm (Proposed)

1: **for** $i = 1, \ldots, f$ **do**
2: $\quad \mathbf{w_i} \leftarrow \texttt{pls\_composite}(\mathbf{X}, \mathbf{Y}, n_i)$
3: $\quad \mathbf{w_i} \leftarrow \mathbf{w_i}/||\mathbf{w_i}||$
4: $\quad \mathbf{t_i} \leftarrow \mathbf{X}\mathbf{w_i}/||\mathbf{X}\mathbf{w_i}||$
5: $\quad \mathbf{X} \leftarrow \mathbf{X} - \mathbf{t_i}\mathbf{t_i}^\mathsf{T}\mathbf{X}$
6: $\quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t_i}\mathbf{t_i}^\mathsf{T}\mathbf{Y}$
7: **end for**

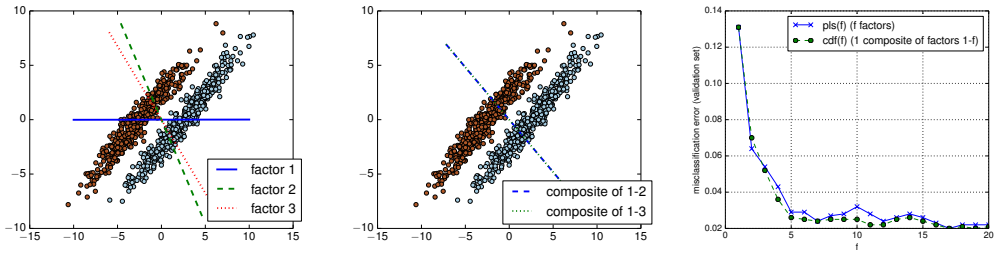

Figure 4.3: Motivating examples. **Left**: Example of how initial PLS dimensions are influenced by input feature covariance. A 3-dimensional dataset is generated by sampling from a Gaussian distribution with standard deviations of $[.5, 4, 1]$ on the diagonal, rotating by 45 degrees in the x-y plane, and shifting the class means apart. The plots show the projection of all points on the x-y plane. The first PLS factor is visibly influenced by the principal axis, causing confusion between the two classes when points are projected onto the factor. The second factor corrects for this, and the third reverses some of the correction. **Middle:** the composites of the factors on the left. In this toy example two factors are enough to create a discriminative composite (a single projection vector). **Right:** comparison between classification error obtained by QDA on f PLS factors (an f-dimensional subspace) versus the composite of the first f factors (a 1-dimensional subspace); trained and evaluated on the *gisette* training and validation subsets, respectively.

$\texttt{pls\_composite}(\mathbf{X}, \mathbf{Y}, f)$. The only parameter for PLS is the number of factors f needed for regression or feature extraction, and is usually set by cross-validation.

4.3 Composite Discriminant Factors

While PLS has been successfully used to select subspaces that are discriminative for classification tasks, the factors that are chosen are not very compact. For example, in Figure 4.3 the initial factor is affected by the covariance of the data $\mathbf{X}$, which in this

case is not informative for discrimination. By extracting sufficient factors, PLS eventually overcomes this problem. The middle plot shows the *composite* projection vector $\mathbf{B} = \texttt{pls\_composite}(\mathbf{X}, \mathbf{Y}, f)$, a single vector computed as a linear combination of the f PLS factors (which is why we call it a *composite*) by PLS regression. It is evident that because PLS regression maps from the latent space to the class indicator, the composite is able to encode the discriminative direction in a single vector. The two plots on the left of Figure 4.3 are toy examples, but the pattern appears in real data as well–the third plot is only one of many examples where a single composite matches and even outperforms Quadratic Discriminant Analysis (QDA) applied to the f factors from which the composite is computed. These examples suggest that while a large set of latent factors that maximize covariance may lead to good discrimination, it is possible to achieve the same results with a more compact set of factors, motivating our approach, Composite Discriminant Factors (CDF).

Just as the PLS algorithm alternates between computing a factor and deflating the data matrices, we can iterate CDF as well, in this case between computing a composite and deflating by that composite. It is easy to show that as long as the composite is a linear combination of the rows of the deflated $\mathbf{X}$, the properties of the PLS deflation process are satisfied, i.e., $\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}$ and $\mathbf{T}^\mathsf{T}\mathbf{T} = \mathbf{I}$. The composite $\mathbf{B}$ is in the row span of $\mathbf{X}$, since it is a linear combination of factors which are each in the row span of $\mathbf{X}$. CDF is parameterized by a length f list $(n_1, n_2, \ldots, n_f)$ of the number of factors n_i to use for the $i^{\texttt{th}}$ composite, and proceeds in a similar fashion to PLS, as shown in Algorithm 4.2.

The parameter space is now much larger than that of PLS, each parameter list representing a linear subspace obtained from the row span of $\mathbf{X}$, and is depicted visually

as a tree in Figure 4.4. The root node corresponds to the original input data $\mathbf{X}$, edges correspond to candidate composites, and child nodes correspond to parent nodes deflated by the composite along the edge. In Figure 4.4 we denote PLS and CDF, along with their parameters, by $\texttt{pls}(f)$ and $\texttt{cdf}(n_1, \ldots, n_f)$, respectively. It is easy to see that $\texttt{cdf}(n_1 = 1, \ldots, n_f = 1) = \texttt{pls}(f)$, so PLS can be represented in the CDF parameter space. Because this parameter space is so large, we propose a best-first search algorithm for the CDF subspace that is optimal for a classification task, potentially with some bounded depth. The search process proceeds by opening children of the node that has so far yielded the best cross-validation score. Here, "opening" a node means that CDF with the corresponding parameters is instantiated and evaluated by cross-validation. Once the search terminates, the parameters corresponding to the node with the best cross-validation score are chosen. Alternatively, to take advantage of parallelism and allow training on a cluster, we can explore all parameters given a maximum number of composites and factors per composite using standard cross-validation. For example, if we consider up to 2 composites with up to 3 pls factors per composite, we would predict the cross-validation error of 12 models: $\texttt{cdf}(1)$, $\texttt{cdf}(2)$, $\texttt{cdf}(3)$, $\texttt{cdf}(1,1)$, $\texttt{cdf}(1,2)$, $\texttt{cdf}(1,3)$, $\texttt{cdf}(2,1)$, $\texttt{cdf}(2,2)$, $\texttt{cdf}(2,3)$, $\texttt{cdf}(3,1)$, $\texttt{cdf}(3,2)$, $\texttt{cdf}(3,3)$. Contrast this with cross-validation for a PLS model with at most three factors, where we need to choose between three models: $\texttt{pls}(1)$, $\texttt{pls}(2)$, or $\texttt{pls}(3)$. Training PLS or a single-composite CDF is fast (similar to training a linear SVM model), but it is easy to see that training time could increase significantly with the number of composites; while it is sometimes acceptable to sacrifice time during training, we have found empirically that CDF generally achieves its peak performance using up to two composites.

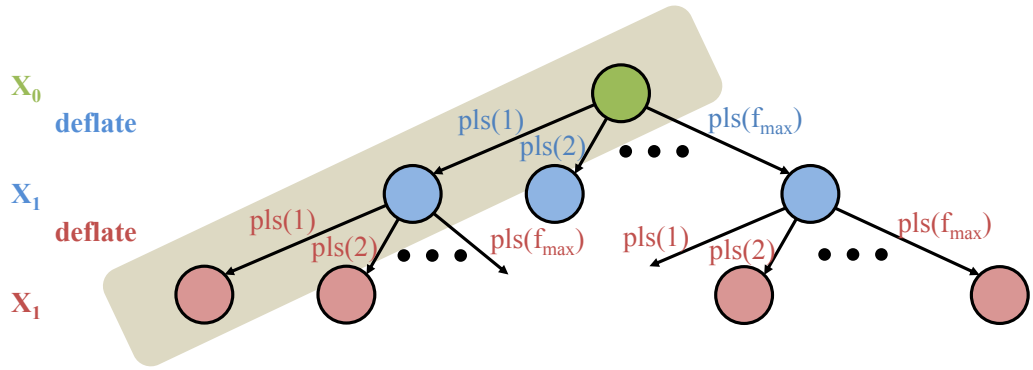

Figure 4.4: Visualization of CDF parameter space. The root signifies the input data matrix, and each level below the root corresponds deflation by an additional composite. The highlighted path corresponds to the original PLS algorithm, so CDF should at least match PLS performance if model selection is sufficiently good.

Although CDF composites have so far been obtained by nested iterations of PLS, other projection directions can be considered as well. For example linear SVM weight vectors are linear combinations of support vectors, so they are also in the row span of $\mathbf{X}$. In this case, a copy of original uncentered $\mathbf{Y}$ indicator matrix is needed at line 2 of Algorithm 4.2, instead of the deflated $\mathbf{Y}$ matrix used for PLS. Other approaches, such as CPLS or CPPLS [63] could be used to propose projection directions. We will focus on CDF with composites obtained by PLS in this paper, leaving other methods for future work.

4.4 Experiments

4.4.1 Action Recognition: UCF50

We evaluate our method on the task of multiclass action recognition using the UCF50 dataset [48], which consists of realistic YouTube videos that span 50 action categories. We represent each video as a 14965 dimensional vector of ActionBank fea-

tures [43], and we predict which of the 50 categories each video belongs to. We perform 5-fold group-wise cross-validation as done in [43] and compare the average accuracy of the following algorithms: **1) linear SVM, 1-vs-all** - linear SVM trained on 14965 dimensional feature vectors using a 1-vs-all multiclass scheme. This is the state of the art reported by [43]. **2) linear SVM, 1-vs-1** - a linear SVM is trained for each pair of classes (50 classes, 1225 total pairs). A test sample is assigned to the class with the most votes, where the vote count for class *i* is the number of *i-vs-j* models (49 of them) which classify the sample as class *i*. Ties are resolved recursively by counting votes again but only among classes with tied vote counts. **3) RBF SVM, 1-vs-1** - standard RBF SVM of *libsvm* on the full 14965 dimensional feature vectors. Multiclass classification is performed by 1-vs-1 voting. **4) PLS, 1-vs-1** Pairwise PLS factors are obtained, and each sample is projected onto all factors. RBF SVM with 1-vs-1 voting is applied to the transformed samples for multiclass classification. **5) CDF, 1-vs-1** - Same as 2) but using CDF instead of linear SVM.

Table 4.1 shows the cross-validation accuracy of each approach. The comparison to linear SVM and PLS shows that CDF produces more informative projections for multiclass classification. The comparison against RBF SVM and PLS shows that CDF also outperforms approaches that make use of non-linear classifiers. While *libsvm* uses the same multiclass scheme as CDF (1-vs-1), liblinear uses a 1-vs-all scheme by default, so we also implemented linear SVM with a 1-vs-1 scheme. CDF outperforms linear SVM regardless of multiclass scheme, thus suggesting that the performance improvements over linear SVM are indeed due to CDF and not to the multiclass scheme. Finally, prediction with one CDF composite is just as fast as linear SVM 1-vs-1, and is much faster than

Table 4.1: Group-wise accuracies on the UCF50 Action Recognition dataset.

	Average Accuracy
linear SVM, 1-vs-all [43]	57.90
linear SVM, 1-vs-1	54.48
RBF SVM, 1-vs-1	56.31
PLS, 1-vs-1	55.43
CDF, 1-vs-1	***59.01***

PLS, which has 8-12 factors per class pair and has higher linear projection cost.

4.4.2 Pedestrian Detection: INRIA Pedestrian Dataset

We also evaluate our classifier as part of a human detector on publicly available INRIA Pedestrian Dataset [39], using the modified HOG features proposed in [40]. We evaluate results using the standard PASCAL scheme based on bounding box overlap that produces precision-recall curves and Average Precision (AP) measures, as done by [40] and [64]. State-of-the-art results on this dataset involve improved features (irregular HOG grids, additional channels, etc) [65, 66] and use non-linear classifiers [65, 66], deformable parts [40], or context [67]. However, we focus on rigid templates of HOG features on a regular grid [40], and linear SVM for two reasons: 1) to isolate the contribution of CDF (as opposed to additional machinery such as deformable parts, context, exemplars), 2) simple HOG features and linear SVM are still prevalent as building blocks in state-of-the-art approaches (as is evident in section 4.1.1). We believe that many of these approaches can benefit from the replacement of linear SVM with CDF but leave this for future work.

One of our baseline approaches is Felzenszwalb's DPM root model [40], which consists of two components (two direction-specific detectors) trained using Latent SVM, a framework that automatically adjusts positive bounding boxes during training (from

initial manual annotations) to better align HOG features. Our detector does not model latent variables, but it does use the same HOG parameters as the root model of [40]: windows have a size of 5 $\times$ 15 grid cells, and each grid cell contains 32 features for a total of 2400 features per window. As an initial training set, we randomly sample in scale and translation from the negative training images to obtain two negatives per image. We resize annotated training bounding boxes by their height, and add a vertically flipped duplicate to the positive training set (we learn a single symmetric filter). We then train the classifiers–linear SVM, PLS, and CDF–setting parameters by 20-fold cross-validation. We consider up to 10 PLS factors for both CDF and PLS, and use QDA as the subsequent classifier when considering multiple composites/factors. Once each classifier is trained, we perform sliding window detection, followed by non-maximal suppression and hard-negative mining (up to 50 hard negatives are added each iteration). Multi-scale detection proceeds by sliding window on an image pyramid with 12 intervals per octave.

As Figure 4.5 shows, CDF outperforms the DPM root model, even though CDF uses unmodified positive bounding boxes and trains a single symmetric model, with no latent variables. CDF also outperforms the LDA model of Bharath et al., which achieves an AP of .75 [52] (not drawn in Figure 4.5). In addition, CDF outperforms both PLS and SVM in terms of AP using only a single composite; since overall computational complexity is dominated by high-dimensional linear convolutions at detection time, and PLS has 5 factors (chosen by cross-validation), CDF is just as fast as linear SVM and is 5 times faster than PLS.

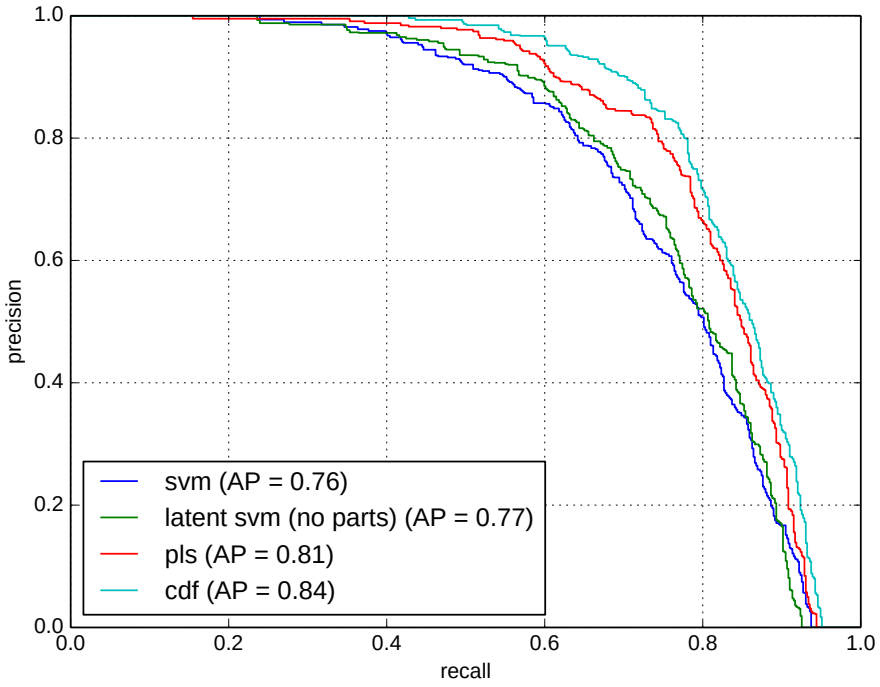

Figure 4.5: INRIA Pedestrian dataset performance. Precision-recall curves comparing CDF to baselines involving rigid templates and linear projection. The CDF curve shown here is obtained using only a single composite (so the classifier is fully linear). The comparison of `cdf` (our approach), to `pls` and `svm` (linear kernel) is fair, i.e., the classifiers are trained using exactly the same approach and input features. The comparison to `latent svm` is unfair to our approach, because of latent positive selection. Comparing `latent svm` to `svm` shows the impact of these additional improvements. Nevertheless, our single-component model without these improvements significantly outperforms `latent svm`.

4.4.3 Vehicle Detection: Google 90° Satellite Dataset

We evaluated the performance of our system on satellite images taken from Google. The dataset consists of 24 high resolution satellite images (RGB) at a resolution of 2048 x 2048 pixels. Train and test sets contain 12 images each with 708 and 1054 vehicles respectively. Training is done by selecting negatives using hard negative mining. We sampled 100 negatives per training image at random locations and orientations giving us

Figure 4.6: Sample vehicle detections the Google 90° dataset. Color represents the confidence of detection, red (high confidence) and blue (low confidence) being the two extremes. ©Google.

1200 initial negatives and 708 positives for training the bootstrap model. We select up to 800 hard negatives in each retraining iteration. We use multi-scale HOG features similar to [41]. Features are calculated for blocks ranging from a size of 12 x 12 pixels to 30 x 66 pixels (window size). Each block yields a 36D feature vector resulting in a total length of 8424. For training, we consider up to 8 PLS factors for PLS and CDF training. Parameters are selected by 20-fold cross validation. Following the example of [46], we use QDA as the classifier after projection. We use horizontal and vertical step size of 1 pixel and we consider every 10 degrees of rotation. Approximately 144 million windows are processed

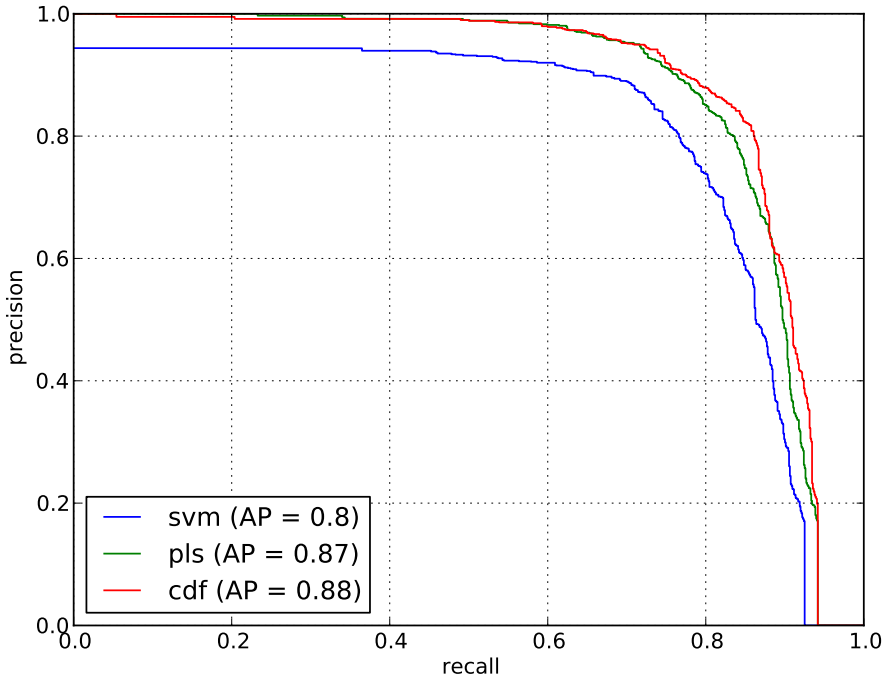

Figure 4.7: Precision-recall curves comparing CDF to the baselines for Google 90° dataset. CDF performs better than SVM in terms of accuracy and is better than PLS in terms of accuracy and efficiency.

per image. Our proposed approach outperforms linear SVM by a large margin and also outperforms PLS in the high recall region Figure 4.7. PLS training chose 8 factors during cross-validation, thus is significantly slower during test time. We also tested PLS by increasing the factors to 20 and still found that classifier with 8 factors was chosen during training. The 2 composites of CDF were obtained by obtaining the composite of first 6 PLS factors, deflating by this composite and then using 1 PLS factor from the deflated data. Figure 4.6 shows detection results on a test image. Figure 4.11 shows the heat map where the detector fires and the corresponding detections after non-maxima suppression. The weight vectors in Figure 4.9 show us that CDF and PLS capture finer details of the vehicle like windows of the car, while SVM just captures contour of the car. To asses the

variation in performance due to the initial random negative set, we repeated the retraining process 10 times, and found that the average precision of the 10 resulting PR curves had a standard deviation of 0.00366.

4.4.4 Vehicle Detection: Google 45° Satellite Dataset

We also evaluate on a vehicle dataset of high resolution 45° oblique view Google satellite images of seven cities: Boston, Houston, Jacksonville, New Orleans, Phoenix, Salt Lake City, and San Francisco. Our goal is to obtain a large dataset of a relatively rigid object (more rigid than pedestrians), allowing us to better evaluate the effectiveness of CDF at modeling appearance statistics that are caused by sources other than non-rigid structural deformations (which generally require deformable parts, exemplars, or other sophisticated machinery on top of the classifier). Images are captured at a 45° angle, so vehicle appearance variation abounds in addition to occlusions (due to tree cover) and image artifacts (due to aerial image stitching). We label vehicles that are occluded, have artifacts, or are larger than a van (e.g., trucks, buses) as ”hard”, and ignore them during training and testing as in [64]. The dataset contains 1104 RGB images at a resolution of 1920 x 1080, divided into a training set (552 images, 6956 vehicles) and a test set (552 images, 8635 vehicles). We train and evaluate the detector as for the pedestrian detection experiments, but we consider multiple rotations (in increments of 10°) instead of multiple scales. We use a window size of 64 x 112 pixels and Felzenszwalb HOG features with a bin and stride of 8. Both PLS and CDF outperform SVM by a large margin as shown in Figure 4.9. CDF yields roughly the same accuracy as PLS, but it does so with only 2

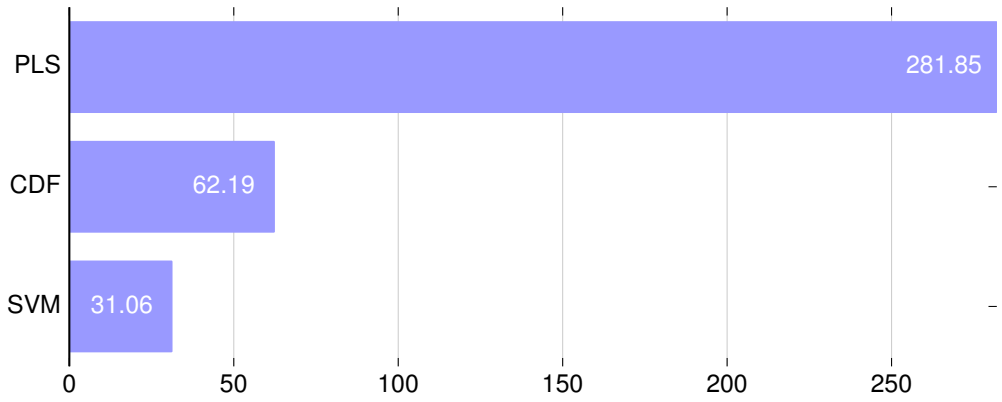

Figure 4.8: Per image test-time sliding windows timings (in seconds) on Google 45°. Timings were taken on an Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz with 6GB RAM. CDF with 2 composites is significantly faster than PLS and is 2 times slower than SVM, but as shown in Figure 4.9, CDF significantly improves over SVM in terms of precision and recall. The timings confirm that the number of linear convolutions determine overall computational expense, since they exhibit a roughly 9:2:1 ratio coinciding with the number of linear projections, for PLS, CDF, and SVM, respectively.

composites, making CDF significantly faster than PLS during test time (see Figure 4.8), by a factor of roughly $9/2 = 4.5$ since sliding window time is dominated by the number of high dimensional linear convolutions. We plan to make the vehicle dataset publicly available.

4.4.5 UCI Machine Learning Repository

To test if CDF outperforms linear SVM and PLS on non-vision datasets, we evaluate the performance of CDF on four standard benchmark datasets from the NIPS 2003 Feature Selection Challenge [31]: *arcene*, *dexter*, *dorothea*, and *gisette*, available from the UCI Machine Learning Repository [49]. Table 4.2 shows the results. While non-linear approaches outperform CDF and our selected baselines, as in the previous experiments, we restrict our attention to linear SVM and PLS+QDA because the main operation in both

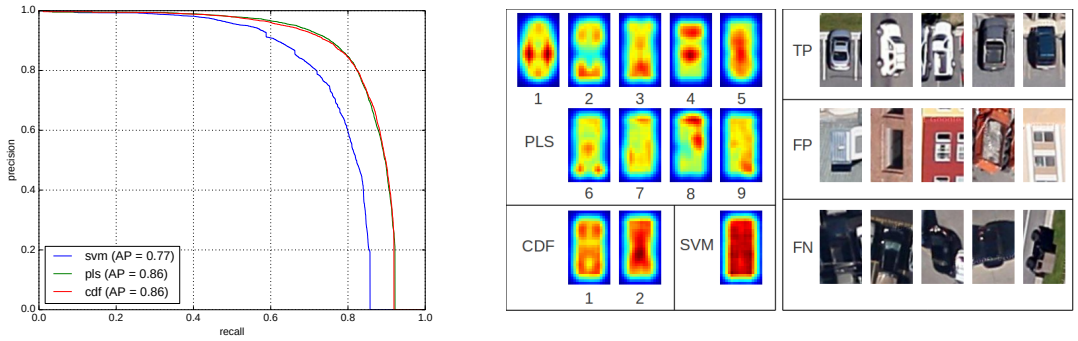

Figure 4.9: Google $45°$ satellite imagery datasets. **Left:** Precision-recall curves comparing CDF to the baselines. **Center:** Backprojection of weight vector magnitudes computed by summing for each pixel the absolute values of the weights it contributed to. PLS captures many variations, but requires 9 factors so is roughly 4.5 times slower than CDF and 9 times slower than linear SVM. Linear SVM requires a single weight vector but captures mostly the contour of the car. CDF captures not only information about the contour of the car, but also the front and rear car windows. **Right:** True positives (TP), false positives (FP) and false negatives (FN) detected by the system. ©Google.

is linear projection, and neither requires the storage of training samples (e.g., as support vectors). We select parameters for CDF, PLS, and SVM using 20-fold cross-validation, selecting up to 11 PLS factors per CDF composite, up to 20 factors for PLS, and a value for the SVM C parameter between 10^{-7} to 10^{7} in powers of 10. We use QDA as the non-linear classifier after PLS or CDF projection. We bound our CDF search at a depth of two (so we find at most two factors), since CDF already matches the performance of SVM and PLS with only one or two composites. Our classification results are relatively invariant to scaling for all but the *arcene* dataset, which we normalize by scaling each feature by its standard deviation (the relative performance between SVM, PLS, and CDF remains fixed even when arcene is not scaled). A noteworthy result is that CDF achieved the reported error rates with 1 composite for *arcene* and *dexter* and 2 composites for *dorothea* and *gisette*. PLS required 6, 4, 6, and 17 factors for the four datasets (in order).

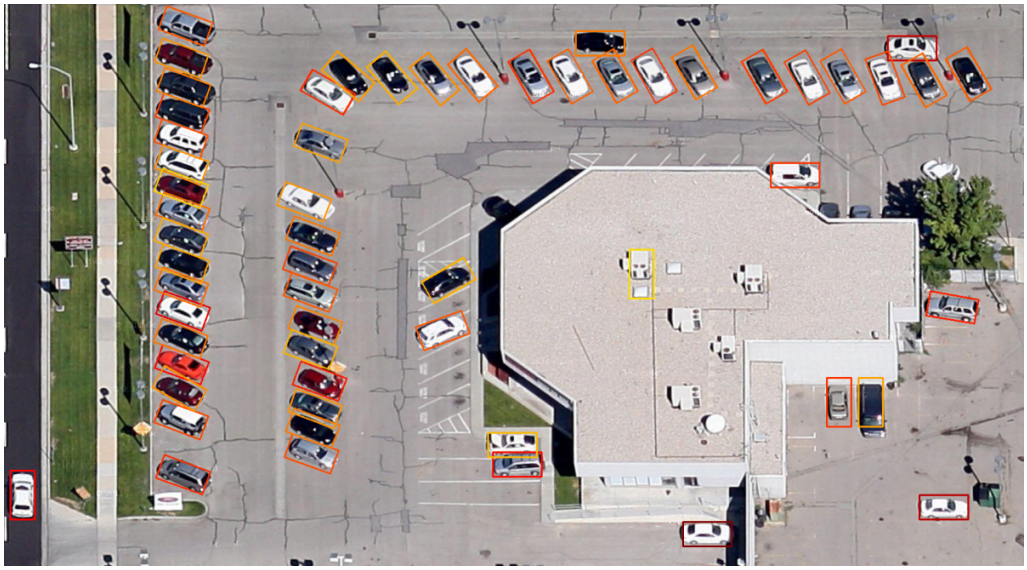

Figure 4.10: Sample vehicle detections the Google 45° dataset. Color represents the confidence of detection, red (high confidence) and blue (low confidence) being the two extremes. ©Google.

Table 4.2: Performance on the UCI ML datasets measured by Balanced Error Rate (BER).

	Arcene	Dexter	Dorothea	Gisette
Dimensionality	10000	20000	100000	5000
Train #pos/neg	44/56	150/150	78/722	3000/3000
Val. #pos/neg	44/56	150/150	34/316	500/500
`svm` BER	.148	.067	.340	.021
`pls` BER	.144	.063	**.145**	.025
`cdf` BER	**.141**	**.060**	.150	**.020**

4.5 Discussion and Future work

We proposed and evaluated a new approach, CDF, which yields surprisingly good performance compared to PLS and SVM, and yields much more compact subspaces than PLS, leading to improved speed at runtime. The improvement is especially noticeable in the vehicle and human detection tasks, as well as on the multiclass action recognition task, suggesting that CDF is a good alternative to linear SVM for many state-of-the-art vision

approaches. Our experiments, however, raise some questions that still need to be investigated. In particular, why do PLS and CDF seem to perform so well against linear SVM? This is still unclear, though we can see that the margin of improvement is much larger for the vision datasets than for the machine learning datasets. A possible explanation is that samples away from the decision boundary have a significant and positive contribution to the projection direction. This can be both an advantage and a disadvantage: more samples contributing to the projection direction can yield a better boundary, but only if the probability mass away from the boundary provides useful information. Other areas that deserve further investigation include the use of additional composite candidates (e.g., SVM), other subsequent classifiers, and extension to a kernel method for applications where kernel methods are practical.

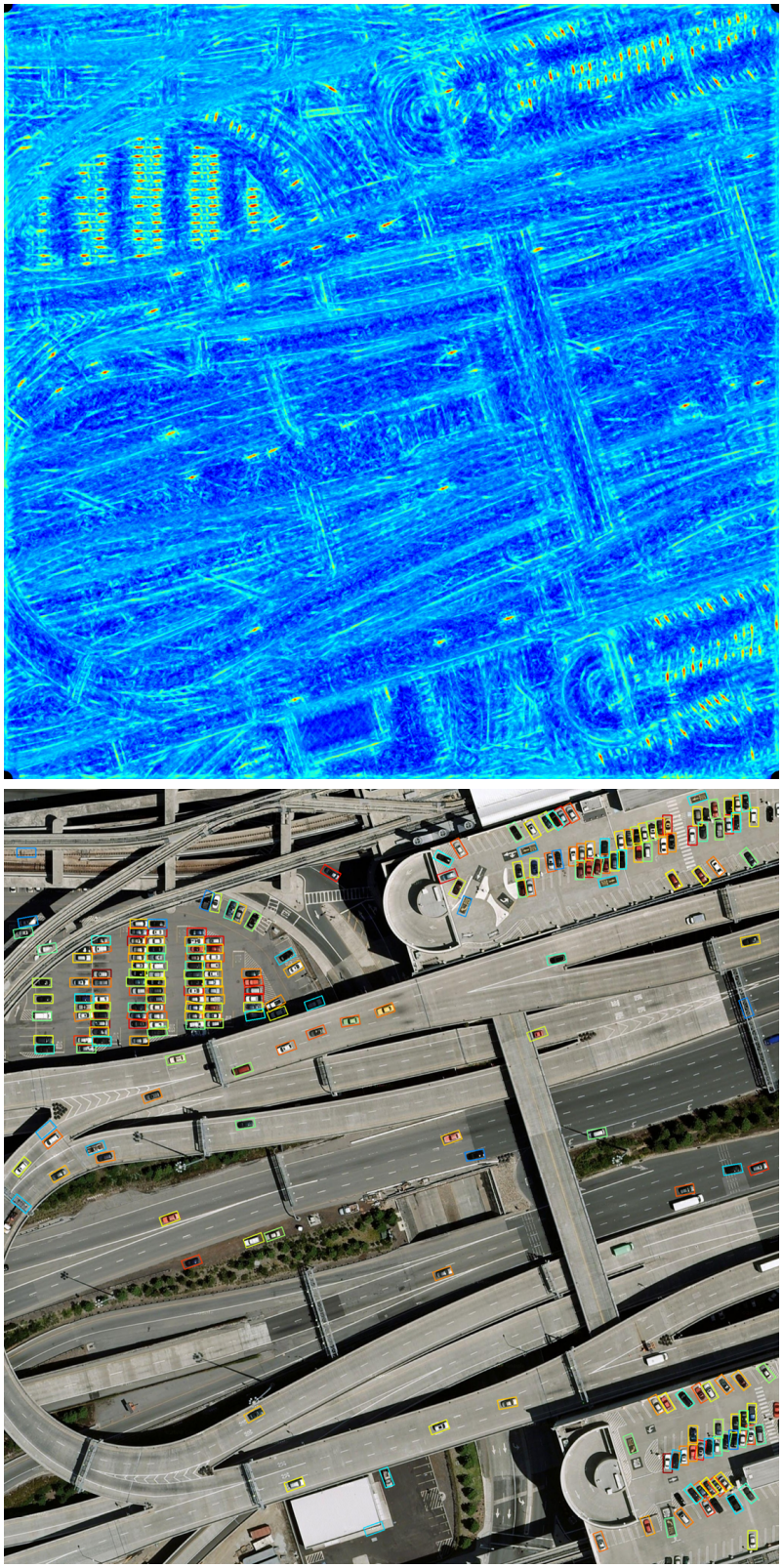

Figure 4.11: Google 90° satellite imagery datasets. **Top:** Detection heat map for CDF. **Bottom:** Corresponding detection bounding boxes after non-maxima suppression. Color represents the confidence of detection, red (high confidence) and blue (low confidence) being the two extremes. ©Google.

Chapter 5: An Improved Deep Learning Architecture for Person Re-Identification

In this work we propose a method for simultaneously learning features and a corresponding similarity metric for person re-identification. We present a deep convolutional architecture with layers specially designed to address the problem of re-identification. Given a pair of images as input, our network outputs a similarity value indicating whether the two input images depict the same person. Novel elements of our architecture include a layer that computes cross-input neighborhood differences, which capture local relationships among mid-level features that were computed separately from the two input images. A high-level summary of the outputs of this layer is computed by a layer of patch summary features, which are then spatially integrated in subsequent layers. Our method significantly outperforms the state of the art on both a large data set (CUHK03) and a medium-sized data set (CUHK01), and is resistant to overfitting. We also demonstrate that by initially training on an unrelated large data set before fine-tuning on a small target data set, our network can achieve results comparable to the state of the art even on a small data set (VIPeR). This work has been published in [68].

5.1 Introduction

Person re-identification is the problem of identifying people in images that have been taken across different cameras, or across time using a single camera. Re-identification is an important capability for surveillance systems as well as human-computer interaction systems. It is an especially difficult problem, because large variations in viewpoint and lighting across different views can cause two images of the same person to look quite different and can cause images of different people to look very similar. See Figure 5.1 for some difficult examples. The problem of re-identification is usually formulated in a similar way to face recognition. A typical re-identification system takes as input two images, each of which usually contains a person's full body, and outputs either a similarity score between the two images or a classification of the pair of images as *same* (if the two images depict the same person) or *different* (if the images are of different people). In this paper, we follow this approach and use a novel deep learning network to assign similarity scores to pairs of images of human bodies. Our network architecture includes two novel layers: a neighborhood difference layer that compares convolutional image features in each patch of one input image to the same features computed on nearby patches in the other input image, and a subsequent layer whose features summarize each patch's neighborhood differences. These novel aspects of our network lead to large improvements over the previous state of the art on the CUHK03 and CUHK01 data sets. We also show that our method is less prone to overfitting on small training sets. We show results on CUHK01 and VIPeR which demonstrate its effectiveness on smaller data sets.

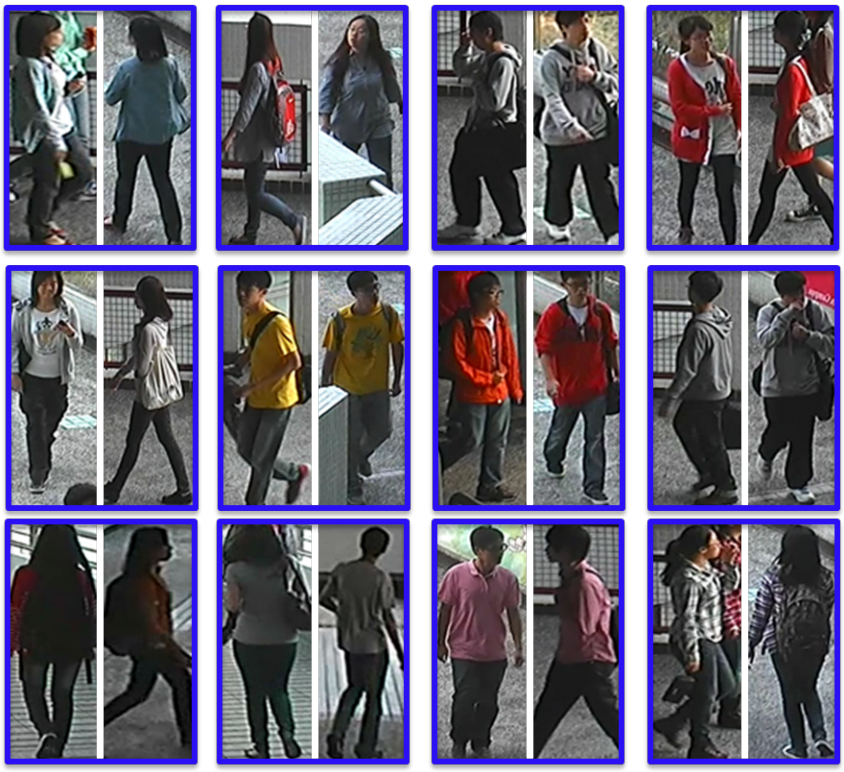

Figure 5.1: Examples of true positives (first row), false positives (second row), and true negatives (bottom row) for our trained network on CUHK03. More results can be found in the supplementary material.

5.2 Related Work

5.2.1 Overview of Previous Re-Identification Work

Typically, methods for re-identification include two components: a method for extracting features from input images, and a metric for comparing those features across images. Research on re-identification usually focuses either on finding an improved set of features ([69–71]), finding an improved similarity metric for comparing features ([72–76]), or a combination of both ([77–79]). The basic idea behind the search for better features is to find features that are at least partially invariant to lighting, pose

and viewpoint changes. Features that have been used include variations on color histograms [72–74, 76–78], local binary patterns [72–74, 77, 78], Gabor features [73], color names [69], and local patches [70]. The basic idea behind metric learning approaches is to find a mapping from feature space to a new space in which feature vectors from *same* image pairs are closer than feature vectors from *different* image pairs. Metric learning approaches that have been applied to re-identification include Mahalanobis metric learning [72], Locally Adaptive Decision Functions [75], saliency weighted distances [74], Local Fisher Discriminant Analysis [77], Marginal Fisher Analysis [77], and attribute consistent matching [78]. Our approach is to learn a deep network that simultaneously finds an effective set of features and a corresponding similarity function.

5.2.2 Deep Learning for Re-Identification

To our knowledge, there have been two previous papers that also used a deep learning approach for re-identification: Yi et al. [80] and Li et al. [79]. In [80], a ”siamese” convolutional network is presented for metric learning. Their network architecture consists of three independent convolutional networks that act on three overlapping parts of the two input images. Each part-specific network consists of two convolutional layers with max pooling, followed by a fully connected layer. The fully connected layer produces an output vector for each input image, and the two output vectors are compared using a cosine function. The cosine outputs for each of the three parts are then fused to get a final similarity score.

Li et al. [79] use a different network architecture that begins with a single con-

volutional layer with max pooling, followed by a patch-matching layer that multiplies convolutional feature responses from the two inputs at a variety of horizontal offsets. (The response to each patch in one input image is multiplied separately by the response to every other patch sampled from the same horizontal strip in the other input image.) This is followed by a max-out grouping layer that keeps the largest response from each horizontal strip of patch-match responses, followed by another convolutional layer with max pooling and finally a fully connected layer and softmax output.

Our architecture differs substantially from these previous approaches. Our network begins with two layers of convolution and max pooling to learn a set of features for comparing the two input images. We then use a novel layer that computes cross-input neighborhood difference features, which compare the features from one input image with the features computed in neighboring locations of the other image. This is followed by a subsequent novel layer that distills these local differences into a smaller patch summary feature. Next, we use another convolutional layer with max pooling, followed by two fully connected layers with softmax output. Along with our new layers which have learnable parameters in them, our network has three convolutional layers as compared to just two in [79] and [80], making our network deeper than previously presented networks for re-identification in the literature. In addition, our network introduces a more powerful way to compare the features learned in the early layers.

Our deep network's re-identification performance exceeds that of all previous approaches on both the large CUHK03 [79] data set and the smaller CUHK01 [81] data set. In addition, even though small data sets can make effective training of large networks difficult or impossible [79], our network performs comparably with the state of the art on

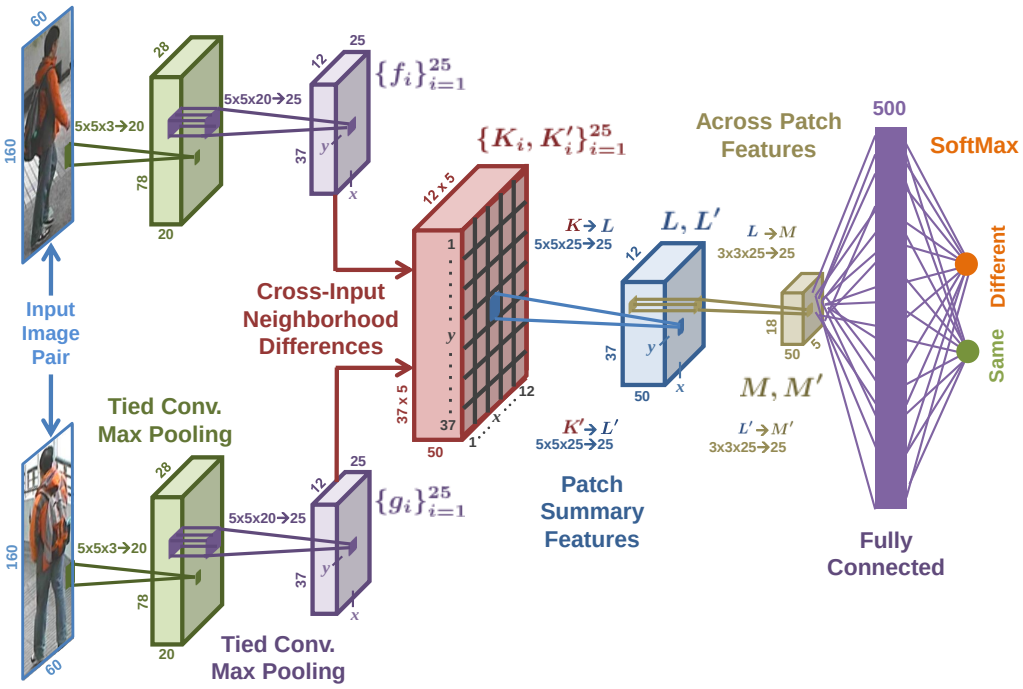

Figure 5.2: Proposed Architecture: Paired images are passed through the network. While initial layers extract features in the two views individually, higher layers compute relationships between them. The number and size of convolutional filters that must be learned are shown. For example, in the first tied convolution layer, $5 \times 5 \times 3 \rightarrow 20$ indicates that there are 20 convolutional features in the layer, each with a kernel size of $5 \times 5 \times 3$. There are 2,308,147 learnable parameters in the whole network. Refer to section 5.3 for more details. [Note that all of the figures in this paper are best viewed in color.]

the much smaller VIPeR data set.

5.3 Our Architecture

In this paper, we propose a deep neural network architecture that formulates the problem of person re-identification as binary classification. Given an input pair of images, the task is to determine whether or not the two images represent the same person. Figure 5.2 illustrates our network's architecture. As briefly described in the previous section, our network consists of the following distinct layers: two layers of tied convolution

with max pooling, cross-input neighborhood differences, patch summary features, across-patch features, higher-order relationships and finally a softmax function to yield the final estimate of whether the input images are from the same person not. Each of these layers is explained in the following subsections.

5.3.1 Tied Convolution

To determine whether two input images are of the same person, we need to find relationships between the two views. In the deep learning literature, convolutional features have proven to provide representations that are useful for a variety of classification tasks. The first two layers of our network are convolution layers, which we use to compute higher-order features on each input image separately. In order for the features to be comparable across the two images in later layers, our first two layers perform tied convolution, in which weights are shared across the two views, to ensure that both views compute the same features. As shown in Figure 5.2, in the first convolution layer we pass input pairs of RGB images of size $60 \times 160 \times 3$ through 20 learned filters of size $5 \times 5 \times 3$. The resulting feature maps are passed through a max-pooling kernel that halves the width and the height of features. These features are passed through another tied convolution layer where we use 25 learned filters of size $5 \times 5 \times 20$, followed by a max-pooling layer that again decreases the width and height of the feature map by a factor of 2. At the end of these two feature computation layers, each input image is represented by 25 feature maps of size 12×37.

5.3.2 Cross-Input Neighborhood Differences

The two tied convolution layers provide a set of 25 feature maps for each input image, from which we can learn relationships between the two views. Let f_i and g_i, respectively, represent the ith feature map ($1 \leq i \leq 25$) from the first and second views. A *cross-input neighborhood differences* layer accumulates differences in feature values around a neighborhood of each feature location across the two views and produces a set of 25 neighborhood difference maps K_i. Since $f_i, g_i \in \mathbb{R}^{12\times37}$, $K_i \in \mathbb{R}^{12\times37\times5\times5}$, where 5×5 is the size of the square neighborhood. Each K_i is a 12×37 grid of 5×5 blocks, in which the block indexed by (x, y) is denoted $K_i(x, y) \in \mathbb{R}^{5\times5}$, where x, y are integers ($1 \leq x \leq 12$ and $1 \leq y \leq 37$). More precisely,

$$K_i(x, y) = f_i(x, y)\mathbb{1}(5, 5) - \mathcal{N}[g_i(x, y)] \tag{5.1}$$

where

$$\mathbb{1}(5, 5) \in \mathbb{R}^{5\times5} \text{ is a } 5 \times 5 \text{ matrix of 1s,}$$

$$\mathcal{N}[g_i(x, y)] \in \mathbb{R}^{5\times5} \text{ is the } 5 \times 5 \text{ neighborhood of } g_i$$

$$\text{centered at } (x, y).$$

In words, the 5×5 matrix $K_i(x, y)$ is the difference of two 5×5 matrices, in the first of which every element is a copy of the scalar $f_i(x, y)$, and the second of which is the 5×5 neighborhood of g_i centered at (x, y). The motivation behind taking differences in

a neighborhood is to add robustness to positional differences in corresponding features of the two input images. Since the operation in (5.1) is asymmetric, we also consider the neighborhood difference map K'_i, which is defined just like K_i in (5.1) except that the roles of f_i and g_i are reversed. This yields 50 neighborhood difference maps, $\{K_i\}_{i=1}^{25}$ and $\{K'_i\}_{i=1}^{25}$, each of which has size $12 \times 37 \times 5 \times 5$. We pass these neighborhood difference maps through a rectified linear unit (ReLu).

5.3.3 Patch Summary Features

In the previous layer, we have computed a rough relationship among features from the two input images in the form of neighborhood difference maps. A *patch summary* layer summarizes these neighborhood difference maps by producing a holistic representation of the differences in each 5×5 block. This layer performs the mapping from $K \in \mathbb{R}^{12\times37\times5\times5\times25} \rightarrow L \in \mathbb{R}^{12\times37\times25}$. This is accomplished by convolving K with 25 filters of size $5 \times 5 \times 25$, with a stride of 5. By exactly matching the stride to the width of the square blocks, we ensure that the 25-dimensional feature vector at location (x, y) of L is computed only from the 25 blocks $K_i(x, y)$, i.e., from the 5×5 grid square (x, y) of each neighborhood difference map K_i (where $1 \leq i \leq 25$). Since these are in turn computed only from the local neighborhood of (x, y) in the feature maps f_i and g_i, the 25-dimensional patch summary feature vector at location (x, y) of L provides a high-level summary of the cross-input differences in the neighborhood of location (x, y). We also compute patch summary features L' from K' in the same way that we computed L from K. Note that filters for the mapping $K \rightarrow L$ and $K' \rightarrow L'$ are different and not tied like

in the first two layers of the network. Both L and L' are then passed through a rectified linear unit (ReLu).

5.3.4 Across-Patch Features

So far we have obtained a high-level representation of differences within a local neighborhood by computing neighborhood difference maps and then obtaining a high-level local representation of these neighborhood difference maps. In the next layer, we learn spatial relationships across neighborhood differences. This is done by convolving L with 25 filters of size $3 \times 3 \times 25$ with a stride of 1. The resultant features are passed through a max pooling kernel to reduce the height and width by a factor of 2. This yields 25 feature maps of size 5×18, which we denote $M \in \mathbb{R}^{5 \times 18 \times 25}$. We similarly obtain across-patch features M' from L'. Filters for the mapping $L \rightarrow M$ and $L' \rightarrow M'$ are not tied.

5.3.5 Higher-Order Relationships

We apply a fully connected layer after M and M'. This captures higher-order relationships by a) combining information from patches that are far from each other and b) combining information from M with information from M'. The resultant feature vector of size 500 is passed through a ReLu non-linearity. These 500 outputs are then passed to another fully connected layer containing 2 softmax units representing the probability of the image pair containing the same person or different people.

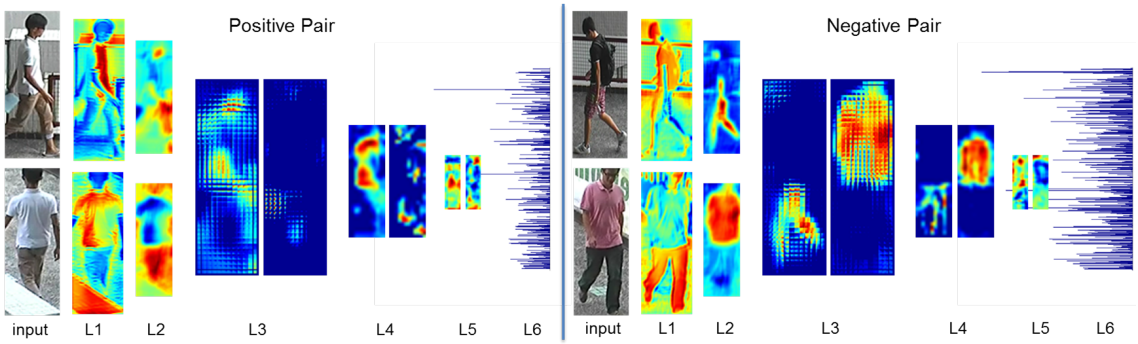

Figure 5.3: Visualization of features learned by our architecture. Initial layers learn image features that are important to distinguish between a positive and a negative pair. Deeper layers learn relationships across the two views so that classification performance is maximized. For details, see Section 5.4.

5.4 Visualization of Features

Figure 5.3 gives a visualization of feature responses at each layer (L1–L6) of the network. The left and right sides of the figure display responses to a positive (same) and negative (different) input pair, respectively. The response maps labeled L1 for the positive pair show the response of one of the 20 features after the first tied convolution layer (see Section 5.3.1). This feature responds strongly to bright white regions of the image, highlighting shirt regions of the person in both views. The maps labeled L1 for the negative pair show the response of a different one of the 20 first-layer features. This feature responds strongly to black regions, highlighting the shirt of the person in view 1 and the pants of the person in view 2. The label L2 indicates feature responses after the second tied convolution layer, which show a pair of feature maps f_i and g_i. The L2 feature shown for the positive pair captures tan and skin-color regions, giving higher responses to the legs, hands and face of the person. Since this is a positive pair, similar parts of the image are highlighted in the two views. In contrast, the L2 feature for the negative pair

activates for different portions of the image across the two views: the legs (pink shorts and pinkish skin) of the person in view 1, versus the torso (pink shirt and pinkish arms) of the person in view 2.

The images labeled L3 are responses of a feature from the cross-input neighborhood differences layer (see Section 5.3.2). Recall from (5.1) that this layer computes the differences of feature maps from the two views in a neighborhood. The resultant feature difference map is then passed through a ReLu, which clips all negative responses to zero. For a positive pair, ideally the neighborhood difference map should be close to zero. Nonzero values should be small and relatively uniform across the map, mainly because the two feature maps compared are very similar. This is illustrated in the L3 map on the left in the positive pair (one of the K_i maps), which has small but non-zero values distributed throughout the map. The image just to its right, which is its complement K'_i, has values that are all zero or close to zero. For the negative pair, different regions are highlighted by f_i than by g_i, so K_i gives a strong response to legs but zeros elsewhere, whereas K'_i responds only to the person's torso. A similar pattern is observed in the patch summary feature for the negative pair (see Section 5.3.3), labeled L4. Higher-order relations across summarized neighborhood difference maps are captured in L5 (see Section 5.3.4). Finally, L6 shows features after the first fully connected layer (section 5.3.5). Notice that this feature representation of a positive pair is quite different than that of a negative pair. This top-layer feature is discriminative and can be used as input to an off-the-shelf classifier.

Figure 5.4 shows a visualization of the weights learned by the first tied convolution layer. The weights shown were learned on the CUHK03 data set. In addition to capturing

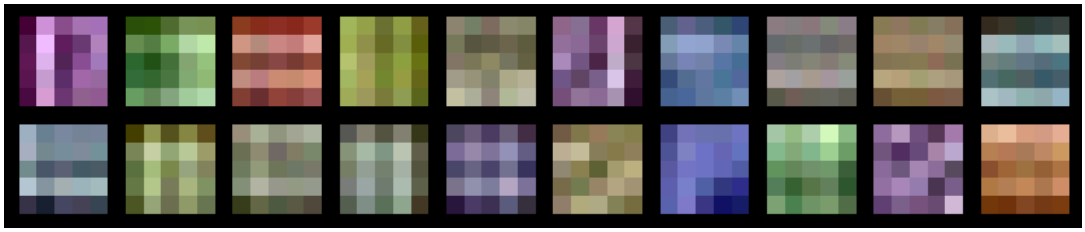

Figure 5.4: Visualization of the weights learned in the first tied convolution layer. Each filter has size $5 \times 5 \times 3$.

some low-level texture information, several of these learned filters exhibit a strong color specialization.

5.5 Other Deep Architectures

In Figure 5.7 (right), we compare our presented network with other variations to gain insights into how much each of our network's novel features contributes to its overall performance. We describe some of these variations here.

Element-wise difference: This architecture illustrates the benefit of comparing with the neighborhood in cross-image comparisons. In this architecture, we perform two layers of tied convolution followed by max pooling. We then compute a cross-input element-wise difference (rather than cross-input neighborhood differences) of the corresponding feature maps. This difference is passed through another layer of convolution followed by a fully connected layer and then softmax.

Disparity-wise convolution: This architecture illustrates the benefit of computing patch summary features. As in our presented network, this architecture performs two tied convolutions followed by max pooling, after which cross-input neighborhood differences are computed. But in this network, the 50 neighborhood difference maps of size $\mathbb{R}^{12\times37\times5\times5}$,

are rearranged to give 25 groups of 50 feature maps, where each feature map has size $\mathbb{R}^{12\times37}$. A convolution is then applied to each of these groups. This is then passed through a fully connected layer and then softmax. Rather than explicitly summarizing neighborhood differences, this architecture instead directly learns across-patch relationships. Figure 5.5 shows the layers which differ from our proposed network.

4 layer convnet: This architecture illustrates the benefit of having a total of four convolutional layers, rather than two as in previous deep approaches to re-identification. We implemented a siamese type network similar to [80], but built the network with 4 layers of convolution rather than 2.

FPNN: We also created our own implementation of FPNN [79] to facilitate comparisons with their results.

5.6 Training the Network

We pose the re-identification problem as binary classification. Training data consist of image pairs labeled as positive (same) and negative (different). The optimization objective is average loss over all pairs in the data set. As the data set can be quite large, in practice we use a stochastic approximation of this objective. Training data are randomly divided into mini-batches. The model performs forward propagation on the current mini-batch and computes the output and loss. Backpropagation is then used to compute the gradients on this batch and network weights are updated. We perform stochastic gradient descent [82] to perform weight updates. We start with a base learning rate of $\eta^{(0)} = 0.01$ and gradually decrease it as the training progresses using an inverse policy:

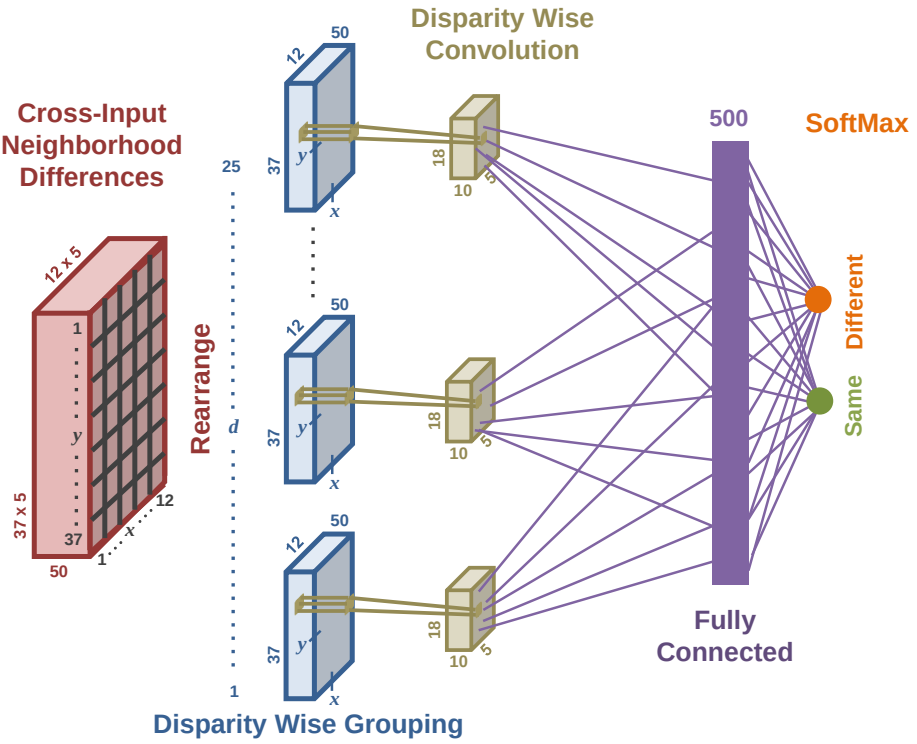

Figure 5.5: Disparity-wise Convolution: The initial layers are the same as our proposed architecture. Only layers that differ are shown. First, cross-input neighborhood differences are rearranged into disparity-wise groups. Each group shows feature differences at offset d. For instance, group 1 contains the values from position $(1, 1)$ of every 5×5 block in the grid of cross-input neighborhood differences, and group 25 contains the values from position $(5, 5)$ of every block in the grid. Convolution is then applied on each group separately. This is then passed through a fully connected layer and then softmax. Instead of explicitly summarizing neighborhood differences, this architecture directly learns across-patch relationships.

$\eta^{(i)} = \eta^{(0)}(1+\gamma \cdot i)^{-p}$ where $\gamma = 10^{-4}$, $p = 0.75$ and i is the current mini-batch iteration. We use a momentum of $\mu = 0.9$ and weight decay $\lambda = 5 \times 10^{-4}$. With more passes over the training data, the model improves until it converges. We use a validation set to evaluate intermediate models and select the one that has maximum performance. Figure 5.6 shows the performance on the validation set as a function of mini-batch iterations on the CUHK03 labeled data set. Each mini-batch contains 100 training samples.

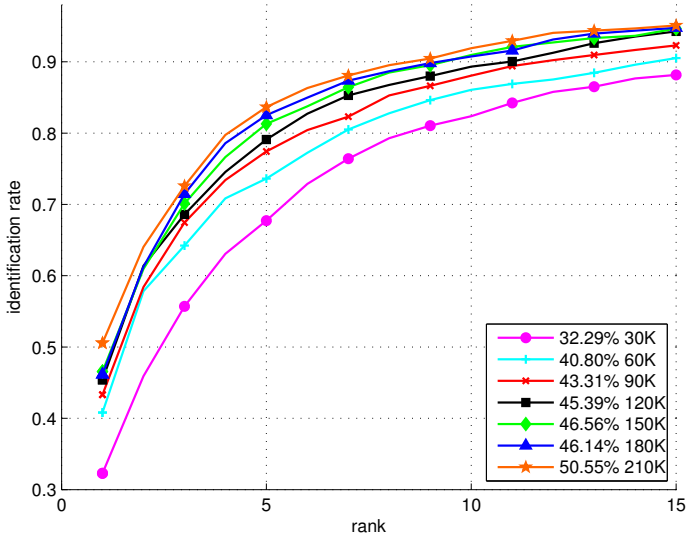

Figure 5.6: Performance on validation set as a function of mini-batch iterations on the CUHK03 labeled data set. In each row of the legend, the first number is the rank-1 accuracy, and the second is the number of mini-batch iterations.

5.6.1 Data Augmentation

There are not nearly as many positive pairs as negative pairs, which can lead to data imbalance and overfitting. To reduce overfitting, we artificially enlarge the data set using label-preserving transformations [83]. We augment the data by performing random 2D translation, as also done in [79]. For an original image of size $W \times H$, we sample 5 images around the image center, with translation drawn from a uniform distribution in the range $[-0.05H, 0.05H] \times [-0.05W, 0.05W]$. For the smallest data set (see Section 5.7.3), we also horizontally reflect each image.

5.6.2 Hard Negative Mining

Data augmentation increases the number of positive pairs, but the training data set is still imbalanced with many more negatives than positives. If we trained the network with this imbalanced data set, it would learn to predict every pair as negative. Therefore, we randomly downsample the negative set to get just twice as many negatives as positives (after augmentation), then train the network. The converged model thus obtained is not optimal since it has not seen all possible negatives. We use the current model to classify all of the negative pairs, and identify negatives on which the network performs worst. We retrain the fully connected (top) layer of the network using a set containing as many of these difficult negative pairs as positive pairs[1].

5.6.3 Fine-tuning

For small data sets that contain too few positives for effective training, we initialize the model by training on a large data set. After hard negative mining on the large set, the parameters of the converged model are then adapted on the new, small data set. For this new network learning, we begin stochastic gradient descent with learning rate $\eta^{(0)} = 0.001$ (which is $1/10$th the initial pre-training rate).

5.7 Experiments

We implemented our architecture using the Caffe [84] deep learning framework, adapting various layers from the framework and writing our own layers that are specific

[1] We also tried retraining the entire network, but retraining just the top layer was more effective.

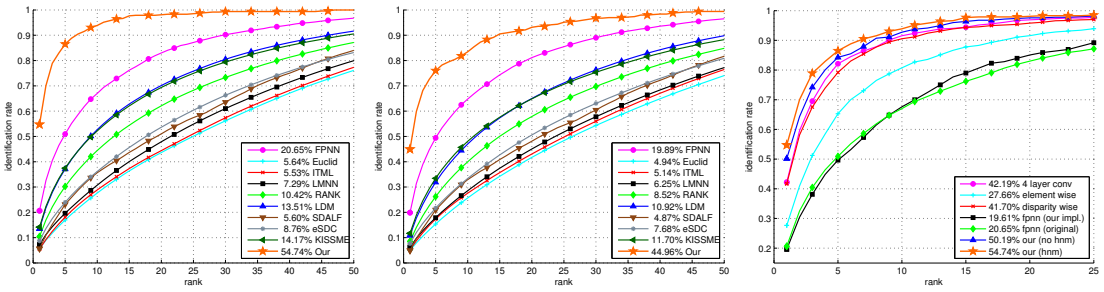

Figure 5.7: CMC curves on CUHK03 data set: a) and b) compare our method with previous methods on CUHK03 labeled and detected, respectively. Rank-1 identification rates are shown in the legend next to the method name. Our method beats the state of the art by a large margin. c) Comparison of our method with our own variations of deep architectures on CUHK03 labeled. Out of the shown methods, only FPNN is previously mentioned in the literature. See section 5.7.1 for details.

to our architecture. Network training converges in roughly 12–14 hours on NVIDIA GTX780 and NVIDIA K40 GPUs.

We present a comprehensive evaluation of our approach by comparing it to the state-of-the-art methods on various data sets. The experiments are conducted with five random splits, and all of the Cumulative Matching Characteristics (CMC) curves are single-shot results. We first report results on the largest re-identification data set in the literature, CUHK03 [79]. Next we report results on the CUHK01 data set [81], using two distinct settings: a) 100 identities in the test set, as reported in [79], and b) 486 identities in the test set, as reported in most previous work on the CUHK01 data set. We also report results on the VIPeR data set [85]. VIPeR and the 486-identities setting of CUHK01 are small data sets, making it difficult for deep networks to learn their parameters without overfitting. Because of this, [79] does not report results on these two data sets.

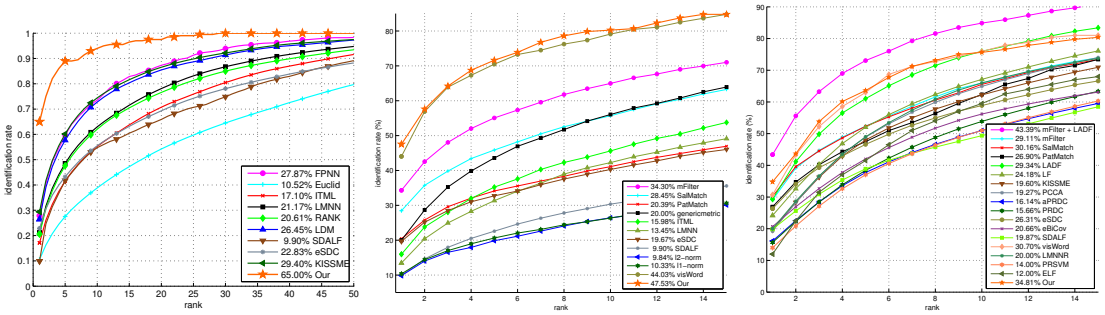

Figure 5.8: CMC curves on CUHK01 and VIPeR data sets: a) CUHK01 data set with 100 test IDs: Our method outperforms the state of the art by more than a factor of 2. b) CUHK01 data set with 486 test IDs: Our method outperforms all previous methods on this data set with this protocol, as well. c) VIPeR: Our method beats all previous methods individually, although a combination of mFilter + LADF performs better than us. Note that (b) and (c) are especially challenging for deep learning methods since there are very few positive pairs. See Sections 5.7.2 and 5.7.3 for more details

5.7.1 Experiments on CUHK03

The CUHK03 data set contains 13,164 images of 1,360 pedestrians, captured by six surveillance cameras. Each identity is observed by two disjoint camera views. On average, there are 4.8 images per identity in each view. This data set provides both manually labeled pedestrian bounding boxes and bounding boxes automatically obtained by running a pedestrian detector [86]. We report results on both of these versions of the data (labeled and detected).

Following the protocol used in [79], we randomly divide 1360 identities into non overlapping train (1160), test (100), and validation (100) sets. This yields about $26K$ positive pairs before data augmentation. We use a mini-batch size of 150 samples and train the network for $210K$ iterations. We use the validation set to design the network architecture.

We compare our method against KISSME [72], eSDC [87], SDALF [88], ITML [89],

logistic distance metric learning (LDM) [90], largest margin nearest neighbor (LMNN) [91], metric learning to rank (RANK) [92], and directly using Euclidean distance to compare features. When using metric learning methods and Euclidean distance, dense color histograms and dense SIFT are used [87]. We also compare against the deep network FPNN [79], which is the current state of the art on this data set.

Figure 5.7 (left) plots the CMC curves of all these methods on the CUHK03 labeled image data set. We outperform the previous deep learning method, FPNN, by a large margin. Our rank-1 accuracy is more than double that of the previous state of the art (54.74% vs. 20.65%). Figure 5.7 (middle) plots performance on the CUHK03 detected image data set. Although the performance of our method on CUHK03-detected is less than on CUHK03-labeled, mainly due to misalignment caused by the detector, our method still greatly outperforms the state of the art (44.96% vs. 19.89%). Figure 5.1 shows some true positive, false positive, and true negative example results of our system.

We also experimented with different rates of dropout after the fully connected layer. Rank-1 accuracy on the validation set for different values of dropout rate are as follows: 46.1% (no dropout), 46.9% (10% dropout), 47.1% (20% dropout), 47.6% (30% dropout), 51.3% (40% dropout) and 50.5% (50% dropout).

We also implemented a variety of other deep network architectures, explained in Section 5.5, to illustrate the benefits of various features of our architecture. We compare with these methods in Figure 5.7 (right). The top two performing methods are our architecture with and without hard negative mining (HNM). Note that other than FPNN, none of the methods in Figure 5.7 (right) has been previously discussed in the literature.

5.7.2 Experiments on CUHK01

The CUHK01 data set has 971 identities, with 2 images per person in each view. We report results for two different settings of this data set: 100 test IDs, and 486 test IDs.

a) **100 test IDs:** In this setting, 100 identities are used for testing, with the remaining 871 identities used for training and validation. This protocol is better suited for deep learning because it uses 90% of the data for training. FPNN [79] uses this setting on this data set. Figure 5.8 (left) compares the performance of our network with previous methods. Our method outperforms the state of the art in this setting by a wide margin, with a rank-1 recognition rate of 65% (vs. 29.40% by the next best method). Notice that the second best method on this data set is KISSME, and not the deep network FPNN. This can be attributed to a decrease in training data as compared to CUHK03, causing FPNN to overfit. In contrast, our method is able to generalize even with this smaller data set.

b) **486 test IDs:** Most previous papers report results on the CUHK01 data set by considering 486 identities for testing. We compare our approach against mid-level filters (mFilter) [71], saliency matching (SalMatch) [76], patch matching (PatMatch) [76], generic metric [81], ITML [89], LMNN [91], eSDC [87], SDALF [88], l2-norm, l1-norm [71], and co-occurrence model using visual word (visWord) [70]. With 486 identities in the test set, only 485 identities are left for training. This leaves only 1940 positive samples for training, which makes it practically impossible for a deep architecture of reasonable size not to overfit if trained from scratch on this data. One way to solve this problem is to use a model trained on CUHK03, then test on the 486 identities of CUHK01. This is unlikely to work well since the network does not know the statistics of the test data set,

and in fact, our model trained on CUHK03 and tested on CUHK01 gave rank-1 accuracy of around 6%, which is far below the state-of-the-art. Instead, we pre-train a network on CUHK03 and adapt it for CUHK01 by fine-tuning (see Section 5.6.3) it on CUHK01 with 485 training identities (non-overlapping with the test set). The performance of the network after fine-tuning for $210K$ iterations increases dramatically, to a rank-1 accuracy of 40.5%. Using this model, we search for hard negatives and use them to retrain the top layer of the network (see Section 5.6.2). After $210K$ iterations, we achieve a rank-1 accuracy of 47.5%, beating the state of the art. See Figure 5.8 (middle) for a comparison with other methods.

5.7.3 Experiments on VIPeR

The VIPeR data set contains 632 pedestrian pairs in two views, with only one image per person in each view. The testing protocol is to split the data set into half, 316 for training and 316 for testing. In addition to the methods listed in section 5.7.2 and 5.7.1, we compare our method against local Fisher discriminant analysis (LF) [93], PCCA [94], aPRDC [95], PRDC [96], eBiCov [97], LMNNR [98], PRSVM [99], and ELF [100]. This data set is extremely challenging for deep network architectures for two reasons: a) there are only 316 identities for training with 1 image per person in each view, giving a total of just 316 positives, and b) the resolution of the images is lower (48×128 as compared to 60×160 for CUHK01). We train a model using the CUHK03 and CUHK01 data sets, then adapt the trained model to the VIPeR data set by fine-tuning on 316 training identities. Since the number of negatives is small for this data set ($90K$), hard negative mining

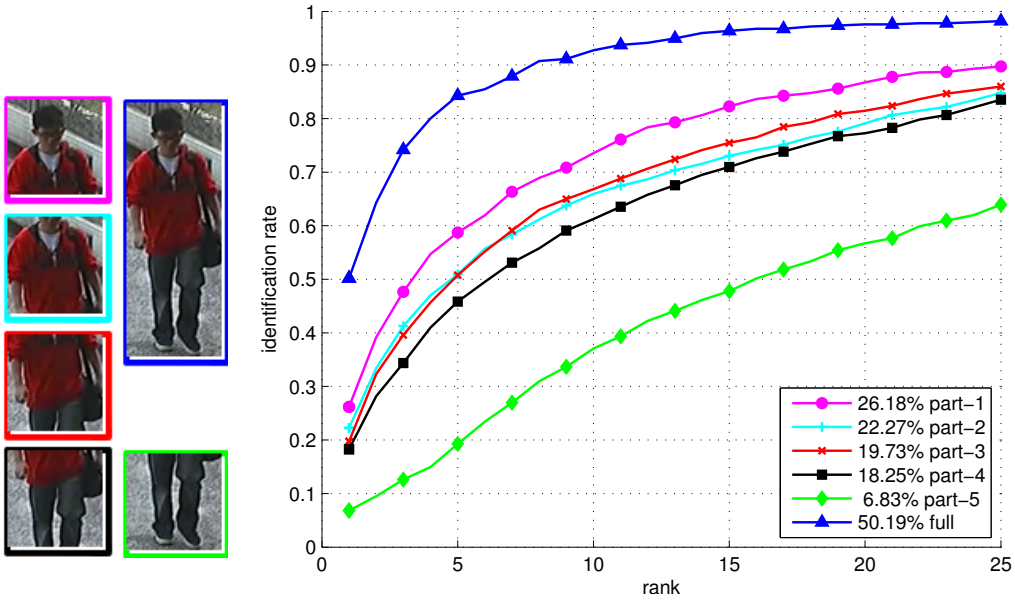

Figure 5.9: Analysis of different body parts: a) Left column shows parts 1 to 4 (from top to bottom). Right column shows full pedestrian image and part 5. b) Shows performance of different parts on the CUHK03 data set. Refer section 5.8.1 for more details.

does not improve results after fine-tuning because most of the negatives were already used during fine-tuning. Figure 5.8 (right) compares performance of our approach with other methods. Our method obtains 34.81% rank-1 accuracy, beating all other methods individually, although a combination of two approaches (mFilter [71] + LADF [75]) performs better than ours with a rank-1 accuracy of 43.4% as reported in [71]. The deep-metric-learning-based method [80] also reports results on the VIPeR data set, with a lower rank-1 accuracy of 28.2%.

5.8 Qualitative Results

Figures 5.10, 5.11, and 5.12 show our system's ranking results on 15 randomly selected identities from the CUHK03 labeled, CUHK01 (100 identities), and VIPeR data sets, respectively. The top 25 results are sorted from left to right.

5.8.1 Analysis of different body parts

To understand the contribution of different body regions to identification, we trained 5 different networks on different body parts, as shown in Figure 5.9 (left). The experiment was performed on the CUHK03 labeled data set, and the performance of each part is shown in Figure 5.9 (right). The best-performing part is the upper region of the body including the face. As we move down the body, the performance decreases, with legs capturing minimum discriminative information. This experiment suggests a direction for future work in which different models can be trained for different parts of the body, and the scores from different part pairs can then be accumulated to reach a final decision. Such a system may be helpful in handling severe occlusions and to identify people in images that have been taken across time (e.g., sitting in one view and standing in the other).

5.9 Conclusion

We have presented a novel deep architecture for person re-identification. We have proposed a novel architecture for finding relationships between two views, by designing cross-input neighborhood differences layer and a subsequent layer that summarizes these differences. We demonstrate the effectiveness of our method by performing a comprehensive evaluation of our approach on various data sets. On the large CUHK03 data set, our method outperforms the state-of-the-art by a huge margin. On the smaller CUHK01 data set (100 test IDs setting), whereas other deep methods overfit [79], our method is able to generalize and produce state-of-the-art results. We also show that models learned by our method on a large data set can be adapted to new, smaller data sets. We demonstrate

this by evaluating our method on two small data sets. On CUHK01 (486 test ids setting), we outperform all previous methods, and on VIPeR, our results are comparable to the state-of-the-art.

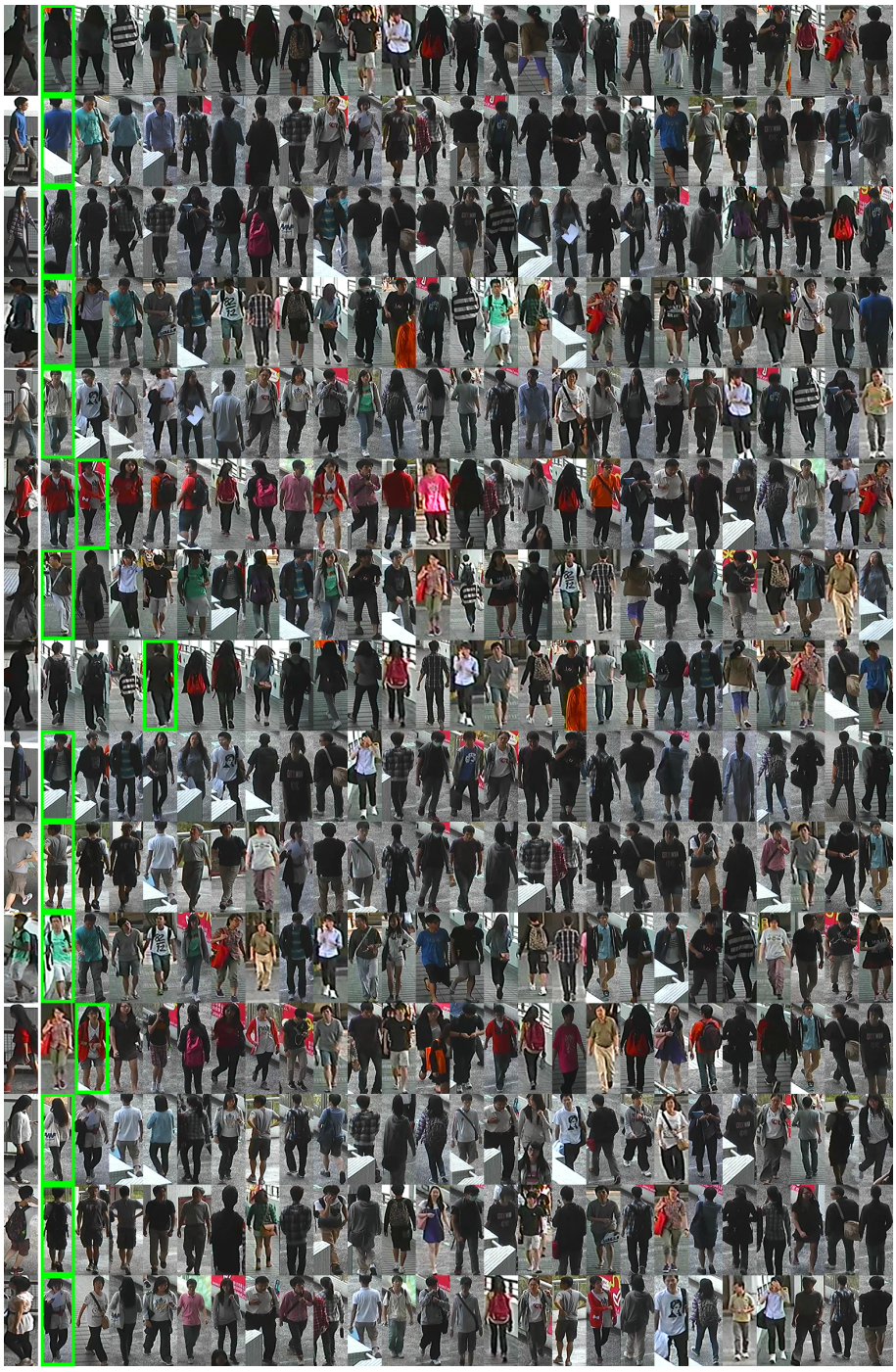

Figure 5.10: Example results on the CUHK03 labeled data set. In each row, the left image is the probe image, and the rest are the top 25 results sorted from left (1) to right (25). The green box indicates the correct match in each row.

Figure 5.11: Example results on the CUHK01 data set (100 identities). In each row, the left image is the probe image, and the rest are the top 25 results sorted from left (1) to right (25). The green box indicates the correct match in each row.

Figure 5.12: Example results on the VIPeR data set. In each row, the left image is the probe image, and the rest are the top 25 results sorted from left (1) to right (25). The green box indicates the correct match in each row.

Chapter 6: Semantic Object Selection

Interactive object segmentation has great practical importance in computer vision. Many interactive methods have been proposed utilizing user input in the form of mouse clicks and mouse strokes, and often requiring a lot of user intervention. In this paper, we present a system with a far simpler input method: the user needs only give the name of the desired object. With the tag provided by the user we do a text query of an image database to gather exemplars of the object. Using object proposals and borrowing ideas from image retrieval and object detection, the object is localized in the target image. An appearance model generated from the exemplars and the location prior are used in an energy minimization framework to select the object. Our method outperforms the state-of-the-art on existing datasets and on a more challenging dataset we collected. This work has been published in [101].

6.1 Introduction

Object segmentation is of great practical importance in computer vision, especially in image editing tasks where operations are restricted to a single object. An important goal in segmentation is to minimize the effort required to select a desired object.

A common approach to object selection is to require the user to provide mouse

(or touch) input to indicate the desired object. Magic Wand [102] requires the user to click on the image and then it selects all pixels with some tolerance. With Intelligent Scissors [103], the user traces the boundary of the object. Graph Cut [104] methods typically require the user to stroke over the object and background. Grabcut [105], a well-known exception to this, rather requires the user to draw a bounding box around the object, and only needs strokes to fix any mistakes. Stroke-based methods have been applied to cosegmentation [106], which also require the collection of similar images with a common object. Matting methods [107–109] require a trimap to be specified. Due to the complexity of natural scenes, overlapping object and background color distributions, and complicated object boundaries, each of these methods often require a lot of tedious user interaction to accurately select the object.

Another approach to segmentation is to perform the selection automatically. Semantic segmentation methods [110, 111] strive to label each pixel in an image with the correct object type. This requires the collection of a large dataset and a known fixed vocabulary, and often needs considerable training time. It is not directed toward an object of user interest, but rather operates on the whole image. Saliency methods [112, 113] are another approach to automatic object selection, where the object which is most visually salient is selected. This usually requires the object to be distinct from the background and quite large in the image. These methods work well if the desired object is in fact the salient object in the image, but this often is not the case.

The goal of this paper is to greatly reduce the user effort required to select an object. This is done by enabling the user to simply name the desired object (Fig. 6.1), either verbally as part of a natural language image processing engine, like PixelTone [114] or

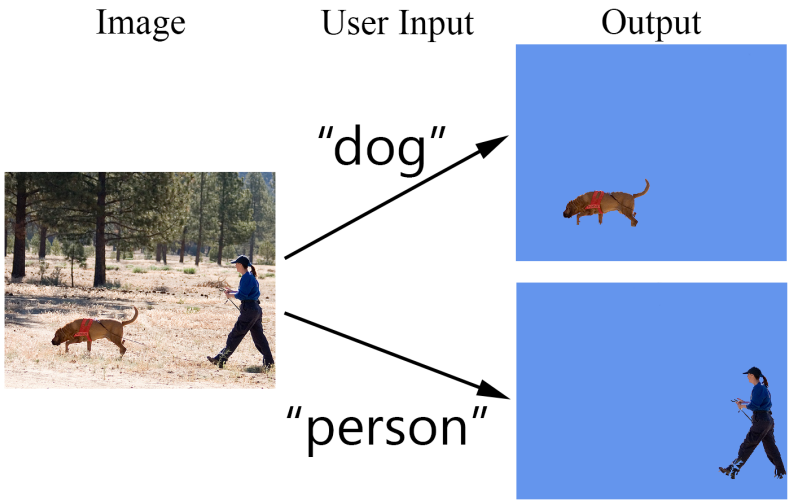

Figure 6.1: Given an image, the user simply provides the name of the object that he/she wants to select. The specified object is segmented by our method without further user input.

by typing it into a search box. For example, if the user makes a PixelTone request such as “Make the *cat* brighter”, our method can be used to identify the cat pixels to be made brighter without any further input from the user (in PixelTone, a user has to paint on the image to mark the cat, name the selection, and then the user can issue semantic editing requests that mention “cat”). With the current proliferation of natural language interfaces for all sorts of tasks (e.g. Siri), we think our method will be very important for advancing image editing via natural language input. This method is more directed than semantic segmentation and does not require a large trained database. Unlike saliency methods, our method can select objects that are small and potentially not salient in the input image as well as objects in images with several salient objects.

We introduce the new problem of Semantic Object Selection in which a user simply specifies the class of the object to select in an image. We propose a solution that scales well with the number of classes, as we do *not* need to train a detector for each class we

wish to recognize. At the core of our system is a novel, robust method using two types of image search for (i) classifying object proposals in the input image as containing the selection class or not and (ii) providing localization information and appearance models for the objects to select. More specifically, a text-based image search, e.g. Google or Microsoft Bing, is used to provide positive images containing instances of the selection class. Negative images unlikely to contain the selection class are also gathered. Object proposals are then computed and used as a query for an image-based search of the positive and negative examples. The object proposals likely to correspond to the desired object are used to compute localization and appearance models that are combined in an energy minimization framework to compute a final selection.

To the best of our knowledge, this tag-based selection system is the first in the literature. Thus we cannot perform direct comparisons. We have compared our method against various other state-of-the-art methods for related problems. We also implemented our own baselines which are competitive by themselves. We have shown results on various classes of the MSRC dataset and the recently introduced Object Discovery dataset. We have also collected more realistic and challenging dataset from imageNet containing dogs. We are comparable to the state-of-the-art on MSRC [115] and beat the state-of-the-art on Object Discovery [1] and our new dataset by a large margin.

6.2 Related Work

While we are unaware of previous work that addresses the problem of selecting a named object without further interaction, there are several lines of work that could be

used to approach this problem.

Saliency methods aim to select the main object in an image by determining which image regions are most "salient". In [112], the saliency is determined by optimizing an energy function which encourages pixels to be salient if they are contained in regions that have high contrast to all other regions. The method in [113] uses similar contrast and location terms, and then computes a binary segmentation by including the computed saliency in a variant of GrabCut [105]. Since these methods do not select object of interest but rather select whichever region stands out, they cannot address the general semantic object selection problem where the object of interest is not the most "salient" object.

Semantic segmentation approaches [110, 111] attempt to automatically segment every object in an image. This could be extended to our problem by labeling every pixel and selecting the pixels corresponding to the named object. There are several drawbacks to this approach. Semantic segmentation methods solve a much large problem than needed, and do not focus on the object of interest. They require a large amount of pre-labeled data and require a predetermined label set that may not contain the desired object label. If the label set contains the desired object, it still may not be found in the image. Many require training classifiers for every label, which for a sufficiently large label set requires excessive computation. Exceptions to this are the non-parametric approaches. These use image retrieval to pull images from the training set for use in transferring labels to the image. For example, in [110] the nearest neighbors in the training set are retrieved and SIFT flow is used to transfer labels to the query image. In [111], globally-similar images are retrieved and the likelihood of each superpixel in query image belonging in each class according to the retrieved set is computed and used in an MRF to compute a segmentation.

There has been much work in object detection, for example [116, 117]. Since object detection localizes a object with a bounding box, it could easily be used to provide a bounding box around a desired object to initialize a segmentation process using a method such as Grabcut [105]. We propose this as a baseline method and compare to it in our results.

Cosegmentation methods [2–5] operate on multiple input images and select in each image a common object. Such methods could be adapted to our problem by performing an Internet search on the query object and performing cosegmentation on the results together with the query image. In fact, Rubenstein et al. [1] propose a method designed to cosegment sets of images collected from an Internet search. It computes a segmentation by optimizing over a function with terms emphasizing sparseness and saliency. Because this method heavily relies on saliency, it is largely restricted to working well on images where saliency methods also work well.

A method which is related to ours is [118]. In this work, a user takes a relatively clean, close-up picture of a product and the goal is to find a similar product to the one in the image. This method iterates between localized image retrieval and selection estimation. The product image database has associated object masks, and the masks of retrieved images are transferred to the query image to estimate the location of the product within the query image. The selection is then computed using a voting scheme based on the image search localizations and is refined using [105]. Note that [118] has a similar goal but simplified input (fairly rigid objects, large and centered in the image with little viewpoint variation).

Unlike [118], our method does not require a database of images with ground truth

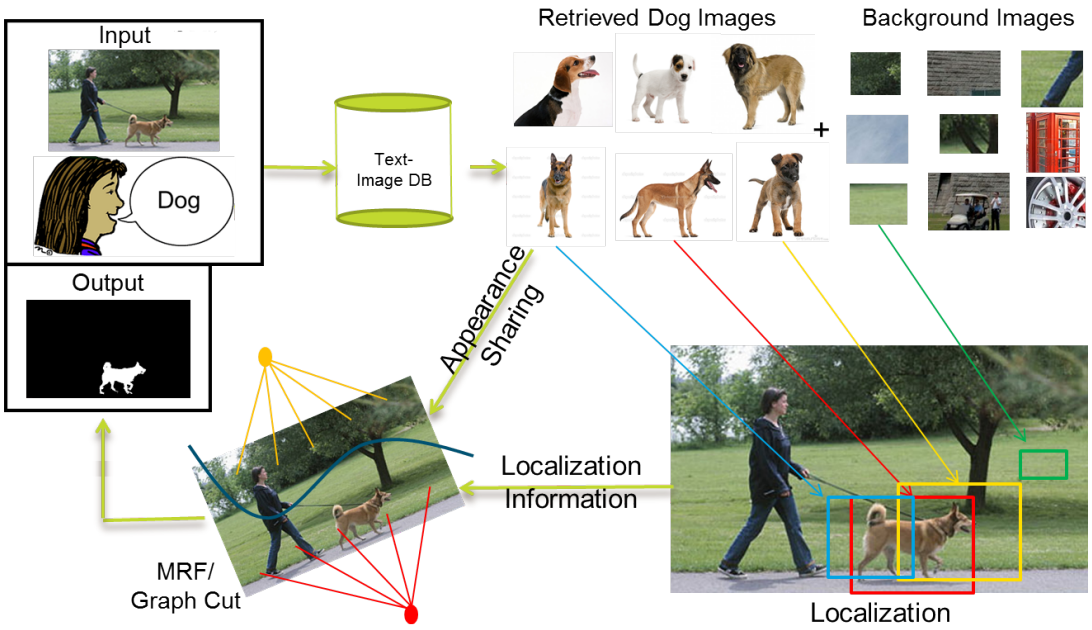

Figure 6.2: Overview of our system: User starts by providing the name of the object to segment. Text-based image search is performed to gather positive exemplars. Positive exemplars along with generalized negatives are then used localize object in the image. This is done with the help of our object retrieval based detection framework. Localization information along with appearance sharing from positive exemplars is used to formulate the segmentation problem as energy minimization. Graph cut is applied on the constructed graph to obtain the desired segmentation.

masks. Our method can handle object classes with larger appearance variations and objects that are less rigid than typical product objects since our method warps the retrieved images to the query image. Our use of object proposals also distinguishes our selection algorithm from the product search work. Clean images where the object of the image is quite large are required in [118] since in cluttered scenes it is difficult to retrieve images that match the object of interest. Our method uses object proposals to make good estimates of where the object may be, which helps avoid background clutter and allows it to handle more general photos of the world.

6.3 Overview

Our method takes as the user input the name of the object as shown in Fig. 6.2. Using this tag we do a text query into an image database to gather exemplars corresponding to the object. Along with positive exemplars we also gather generalized negative examples and put them in an image retrieval database. We then divide our image into object proposals [119]. Each object proposal queries the image retrieval database (using [6]) to validate the presence or absence of the object in a given object proposal.

Once we have found object proposals potentially containing the desired object and their corresponding exemplars, we estimate the location of the object in the corresponding exemplar. We transfer this information onto an object proposal using SIFT flow based image warping [120] to produce a *location prior*. We use this location prior to obtain image specific object and background models. We then combine the image specific appearance model and location prior in a graph cut energy minimization framework.

Note that any state-of-the-art object detectors like DPM or exemplar-SVM [116, 117] followed by our segmentation algorithm cannot be used here since such object detectors usually have an extremely expensive training phase, involving bootstrapping and hard negative mining phases. Since our goal is to deal with large number of object classes, we cannot use pre-trained models. Also in case of DPM, exemplar-based mask transfer cannot be used since various positive examples are clubbed together to build a model. Moreover, in our experiments we show that our method performs better than a DPM-based segmentation algorithm.

Figure 6.3: Positive exemplar database: Objects on white background and exemplars from PASCAL VOC (last 2 columns).

6.4 Localization

Given the tag of the object, our first challenge is to find the location of the object in the image. Our goal is to obtain an object location prior for the target image. We first collect positive exemplars corresponding to the object along with some generalized negative exemplars to build an image retrieval database [6]. We then break the target image into object proposals using [119] and validate the presence of the object in the object proposals. We use SIFT flow to transfer the location associated with validating exemplar areas to the target image. The accumulated location information provides a location prior.

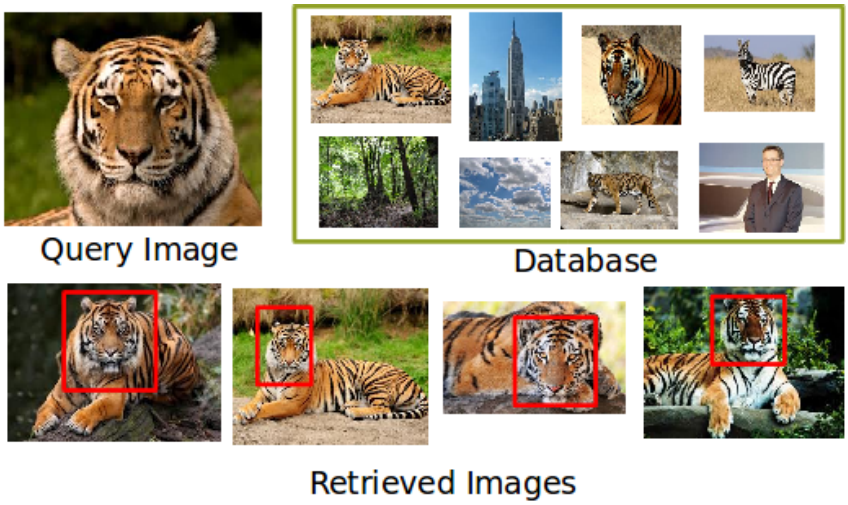

Figure 6.4: Object retrieval with localization: We use object retrieval system of [6] which returns ranked retrieved images along with the bounding box around the matched object.

6.4.1 Exemplar Retrieval Database

Once the tag of the corresponding object is provided by the user, our system needs to learn what our object looks like. We collect positive examples on white background for different classes by querying Google with "<object> on white background". This gave us a large database of positive exemplars (Fig. 6.3). We append this dataset with other publicly available positive sources such as PASCAL VOC. We also use some generalized negative images from INRIA pedestrian dataset. Note that performance increases if the negative exemplars are representative of the background in the target image. Although it is not necessary to accurately represent all background objects, by identifying some likely background regions we can eliminate some potential false positives and improve the localization of the foreground.

Having positive and negative exemplars makes this a typical detection problem. Many state-of-the-art detection systems like [116, 117] try to solve this problem by learn-

ing a classifier between positives and negatives. While such a method is promising, it has its limitations in this scenario as they require a very expensive training step. We instead leverage concepts from image retrieval to obtain the location of the object in an image.

We use the retrieval system from [6] which uses a spatially-constrained similarity measure to handle rotation, scaling, view point change and appearance deformation. The similarity measure is calculated by geometrically aligning SIFT visual words indexing the query and database images; achieving object retrieval and localization simultaneously (Fig, 6.4). We put positive and negative exemplar images into an *exemplar database* for localized search using [6]. This involves computation of SIFT features and creation of an inverted file that stores feature locations for faster voting map generation during retrieval.

6.4.2 Detection via Object Proposal Validation

For object detection, the current state-of-the-art is based on exhaustive search. However, to enable the use of more expensive features and classifiers a selective search is more desired. We use object proposal method proposed in [119]. They have reported a recall of 96.7% with around 1500 windows per image on PASCAL VOC 2007. We divide the target image using object proposals. These object proposals contain desired object, other objects and even background. Our goal is to classify each object proposal as containing the desired object or not. To solve this, we use the exemplar database created in the previous step.

We query each object proposal into the exemplar database. When calculating the voting map for retrieval, we follow the general retrieval framework of [6], i.e., for each

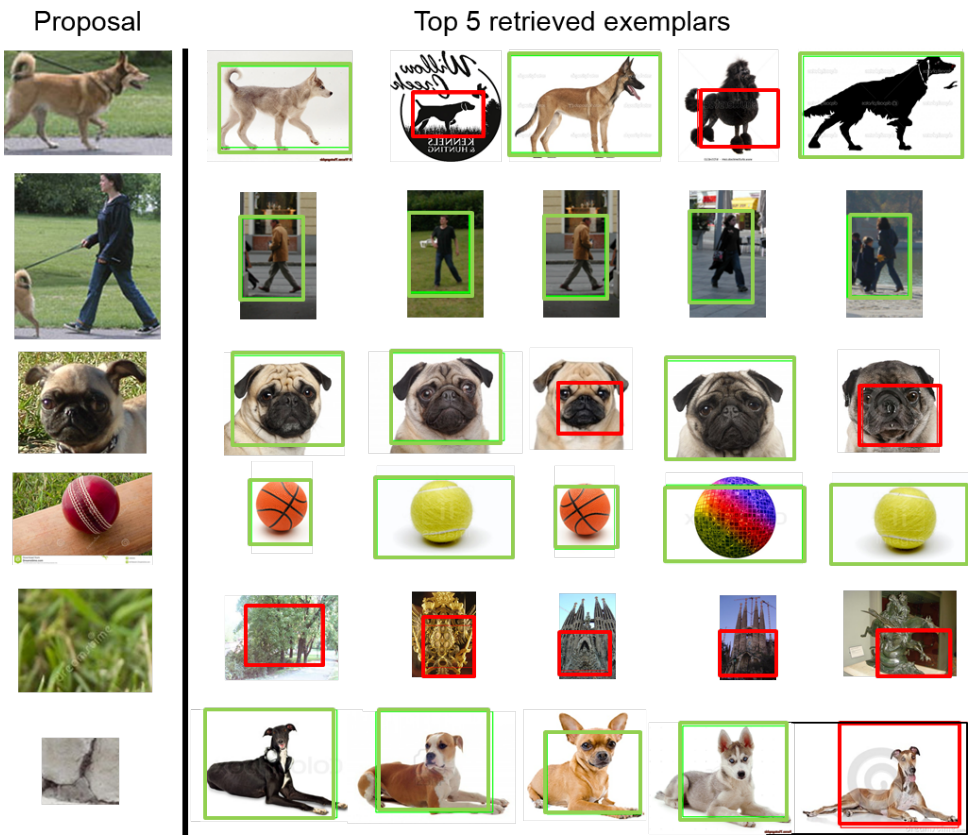

Figure 6.5: Validation: Each row shows an object proposal and its top 5 retrieved exemplars. Retrieved exemplars also contain the bounding box around the matched object. The color of the bounding box specifies whether the exemplar is considered as positive (green) or negative (red). If the box is not centered, e.g. in 1^{st} row 4^{th} exemplar, the exemplar is considered negative. Majority voting decides whether the object proposal contains the specified object or not. The last row shows an example of a false positive where an object proposal is incorrectly validated as a dog. The positive class for each query from top to bottom is dog, person, pug, ball, person and dog.

visual word k in the query, retrieve the image IDs and locations of k in these images through the inverted files. Object center locations and scores are then determined and votes are casted on corresponding voting maps. This results in ranking by similarity score of all the exemplars in the database along with the potential location of the object in the exemplars. We consider the top t exemplars for validation. Recall that each image in the retrieval database is known to be the object of interest or the background. Some of the

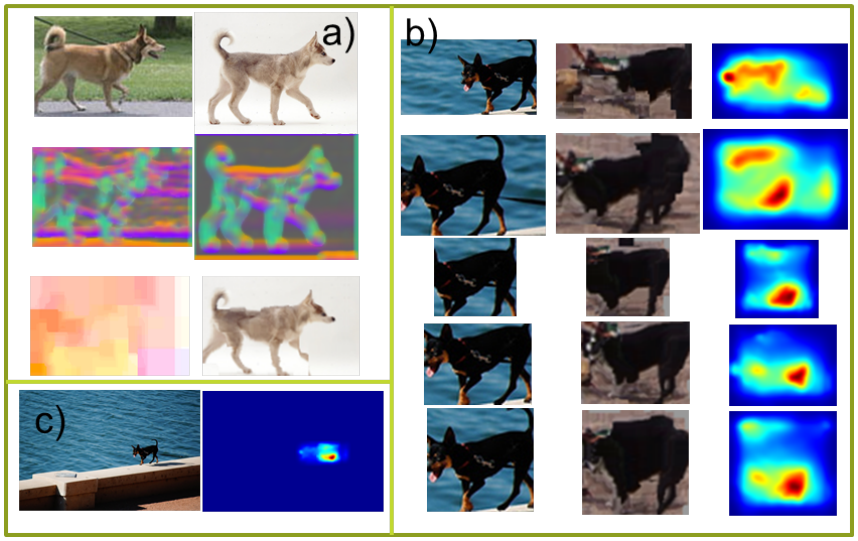

Figure 6.6: Mask Transfer: a) Warping of an exemplar (top right) onto the object proposal (top left). 2^{nd} row shows sift features for object proposal and exemplar. 3^{rd} column shows the sift flow correspondence(left) and warping of exemplar onto the object proposal(right). b) Top 1^{st} column shows object proposals, 2^{nd} column shows best exemplar warped onto the object proposal, and 3^{rd}column shows the saliency mask for the warped exemplars. c) Input image and aggregated location prior.

exemplars in top t belong to the object (tag), while others might belong to background. It might also happen that the exemplar belongs to object (tag), but the bounding box returned by localization is not centered on the exemplar. In this case the exemplar is also considered as negative. From these top t exemplars, majority voting is performed and the object proposal is classified as belonging to the object (tag) if most of the exemplars in top t are positive. See Fig. 6.5.

6.4.3 Location Prior

For each positive retrieved exemplar, we desire to transfer the location of each pixel belonging to the object to the object proposal. Our retrieved images have no associated ground truth masks so we must estimate the location of the object in the retrieved images.

We use saliency to estimate the object location. Since our object proposals are not usually not cluttered and often match to uncluttered retrieved images and since we explicitly search for objects on a white background to collect the retrieval set, we find that saliency works sufficiently well in this constrained use case.

For each object proposal containing the object, its best positive exemplar (according to retrieval score) is considered for segmentation transfer. We obtain soft segmentation mask on the exemplar image by computing saliency map of [121], which gives a score in $[0, 1]$ to each pixel in the exemplar image. We transfer this mask to the corresponding object proposal by SIFT flow warping [120]. Note that many positive object proposals can be shifted versions of the same object and hence their masks can be overlapping on the target image. All masks are aggregated on the target image and re-normalized to lie between $[0, 1]$. Fig. 6.6 shows this process.

6.5 Segmentation

Given the retrieved positive exemplars for each positive object proposal and the location prior, we compute a binary segmentation of the desired object. We pose the segmentation problem in a classic energy minimization framework [105, 122, 123]. Our unary terms consist of an image specific appearance model, an appearance model shared from exemplars, and a location prior. We iteratively minimize the energy, updating our models in each iteration.

Let x_p be the label of the pixel p in the image and $\mathbf{x}$ be the vector of all x_p. The

energy function given the appearance model A and exemplar data X_E can be given by

$$E(\mathbf{x}; A, X_E) = \sum_p E_p(x_p; A, \mathcal{X}_\mathcal{E}) + \sum_{p,q} E_{pq}(x_p, x_q) \tag{6.1}$$

In this the pairwise potential is given by

$$E_{pq}(x_p, x_q) = \delta(x_p \neq x_q) \cdot d(p,q)^{-1} \cdot \exp(-\gamma \|c_p - c_q\|^2), \tag{6.2}$$

where c_p is the color at pixel p. This potential encourages smoothness by penalizing neighboring pixels taking different labels. The penalty depends on the color contrast between pixels, being smaller in regions around image edges (high contrast). We consider an 8-connected pixel grid.

Our unary term is a linear combination of three terms:

$$\begin{aligned} E_p(x_p, \mathcal{A}, \mathcal{X}_\mathcal{E}) = &-\alpha_I \log p(x_p; c_p, \mathbf{A_I}) \\ &-\alpha_{\mathcal{X}_\mathcal{E}} \log p(x_p; c_p, \mathbf{A}_{\mathcal{X}_\mathcal{E}}) - \alpha_M \log M_p(x_p; \mathcal{X}_\mathcal{E}). \end{aligned} \tag{6.3}$$

Each potential $p(x_p; c_p, \mathbf{A})$ evaluates how likely a pixel of color c_p is to take label x_p, according to the appearance model $\mathbf{A}$. The first term uses an image specific image prior $\mathbf{A_I}$. The foreground and background appearances are each separately modeled using a 5 component GMM. The foreground and background are initialized using the location prior; all pixels whose location prior is below some threshold γ_B or above some γ_F are assumed to be background or foreground respectively and are included in the respective

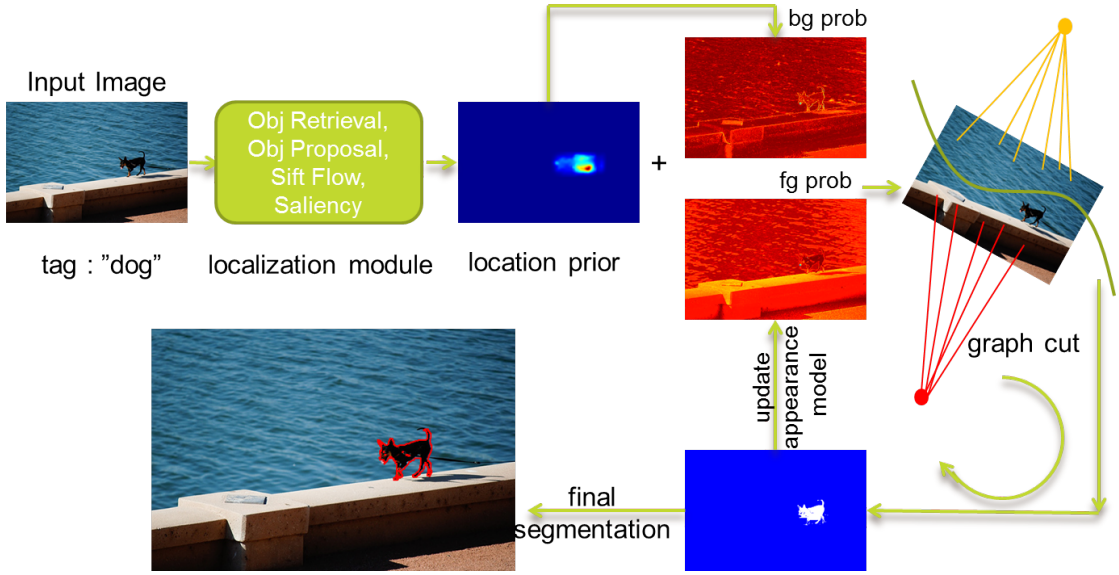

Figure 6.7: Segmentation Framework: Given the input image and the tag, object retrieval based localization is performed to obtain a location prior. Using this location prior, fg and bg probabilities are obtained. These probabilities along with the location prior are used to set the weights of the graph. Graph cut is applied to obtain intermediate segmentation which is used to update our models. After a few iterations a final selection is obtained.

appearance model.

The appearance model $\mathbf{A}_{\mathcal{X}_\mathcal{E}}$ is obtained from the positive exemplars used to compute the location prior. This appearance model is useful in sharing information from exemplars and is particularly useful when segmenting object classes whose appearance does not change over exemplars (particular breed of dog, e.g. brown Labradors).

We obtain the location prior $\mathbf{M_p}$ using exemplar-based image retrieval in the previous step. It is a soft segmentation between $[0, 1]$ and has probabilistic nature. Thus we directly use $\mathbf{M_p}(x_p; \mathcal{X}_\mathcal{E}) = \mathbf{M_p}^{x_p}(1 - \mathbf{M_p})^{1-x_p}$ as a unary potential in Eq. 6.3.

Our segmentation framework, shown in Figure 6.7, is inspired by [105, 122]. A graph is constructed with a node for each pixel and using unary and binary potentials from Eq. 6.1. Graph cut is then used to compute a binary segmentation. The image-

class	ours	[1]	[2]	[3]	[4]	[5]
bike	**55.3**	54.1	43.3	29.9	42.3	42.8
bird	64.6	**67.3**	47.7	29.9	33.2	–
car	**66.8**	66.7	59.7	37.1	59	52.5
cat	**70.7**	66.2	31.9	28.7	37.6	39.4
chair	60.3	**62.2**	39.6	28.7	37.6	39.4
cow	78.5	**79.4**	52.7	33.5	45	26.1
dog	**69.1**	67.5	41.8	33	41.3	–
plane	**58.8**	56.7	21.6	25.1	21.7	33.4
sheep	**81.2**	78.9	66.3	60.8	60.4	45.7
average	**67.3**	66.5	45.0	34.1	42.0	39.9

Table 6.1: Results on MSRC dataset. We compare against Object Discovery [1], Joulin et al. [2], Kim et al. [3], Joulin et al. [4] and Mukherjee et al. [5]. Our method is slightly better or comparable to Object Discovery which is state-of-the-art on MSRC.

specific appearance model is updated given the new foreground. We iterate (5 times) between solving the energy function using graph cut and updating the models.

6.6 Results

We present both qualitative and quantitative results on various datasets. We set $\gamma_B = 0.05$, $\gamma_F = 0.8$, $\alpha_I = 0.6$, and $\alpha_M = 0.4$. The trade off between the unary term and binary term is $\lambda = 50$. While for objects with consistent appearance, the exemplar-specific appearance model can be very useful, we largely tested on objects with a large variation in appearance and thus set $\alpha_{\mathcal{X}_\mathcal{E}} = 0$. To report quantitative results we use Jaccard similarity, i.e. intersection over union of the result and ground truth segmentation.

6.6.1 Results on MSRC Dataset

We report results on the MSRC dataset [115]. We search Google to get objects on white background as positive exemplars, and append this list with PASCAL VOC 2010

Methods	OD airplane	OD car	OD horse	ImageNet dog
Ours	**64.27**	**71.84**	**55.08**	**69.91**
OD [1]	55.81	64.42	51.65	–
Joulin et al. [4]	15.36	37.15	30.16	28.65
Joulin et al. [2]	11.72	35.15	29.53	24.69
DPM+Grabcut	39.47	68.00	50.12	48.24
CEN+Grabcut	37.29	64.96	48.89	34.53
GT+Grabcut (Upper bound)	50.87	80.82	65.99	79.52

Table 6.2: Results on Object Discovery(OD) and ImageNet Dog. On the Object Discovery dataset [1] we perform better than the state-of-the-art by a significant margin. We also compare against our DPM-based segmentation baseline method and outperform it by a significant margin. Note that we beat the upper bound (using ground-truth bounding boxes) on the airplane category. On ImageNet-dog we perform much better than DPM+Grabcut.

training positive examples for each class. 9 of the 14 classes of MSRC are present in PASCAL VOC 2010, we thus compute results on 9 classes of MSRC (around 30 images per class). We compare our performance with [2–5] as reported in [2]. We also compare against the recent Object Discovery work [1] which uses dense correspondences between images to capture the visual variability of common object. This method works well when the object is salient in the image. The cosegmentation methods use all the test images as input to the system while the input to our system is just one label and one image. The quantitative results are given in Table 6.1.

Our method is significantly better than [2–5]. It is also slightly better or comparable to Object Discovery [1]. The closeness in performance is due to the fact that the MSRC dataset contains images with a salient target object and uniform background. This acts as a boon to Object Discovery's approach which is tuned to work well in cases where object is the most salient object in the image. Our approach is a more general approach

which works well in this scenario but is not limited to images with salient objects only. Qualitative results can be found in Figures 6.8, 6.9 and 6.13.

6.6.2 Result on Object Discovery Dataset

To prove our claim that our method is more general and works well when the object is not the only salient object in the image, we test the performance of our method on the Object Discovery dataset. This dataset was introduced in [1] and consists of images downloaded from the Internet. There is large variation in style, color, texture, pose, scale, position and viewing angle. The dataset consists of three classes, car, horse, and airplane, with around 100 images in each category.

In order to further prove effectiveness of our approach we implemented our own baselines. For each image, we initialize a centered bounding box covering 25% of the area of the image and initialize Grabcut [105] using this bounding box. We call this approach CEN+Grabcut. Next, we compared our approach with a detector-based method. We trained discriminative part based detectors [116] on the car, horse, and airplane categories from PASCAL VOC 2010. In order to select a detection threshold we obtained the PR-Curves and selected a threshold corresponding to f1 score. The detection bounding boxes obtained by running the detector are used to initialize Grabcut. We call this method as DPM+Grabcut. Finally, we initialize Grabcut with the ground truth bounding box of the objects in the image. We call this GT+Grabcut. Note that this is an upper bound of a detection plus Grabcut approach. Since [116] uses models trained on PASCAL VOC 2010, we only use PASCAL VOC 2010 training images as positive exemplars so that the

Figure 6.8: Comparison of qualitative results on MSRC for various classes. Left to right, input image, our method, object discovery [1], Joulin et al. [2] and Joulin et al. [4]

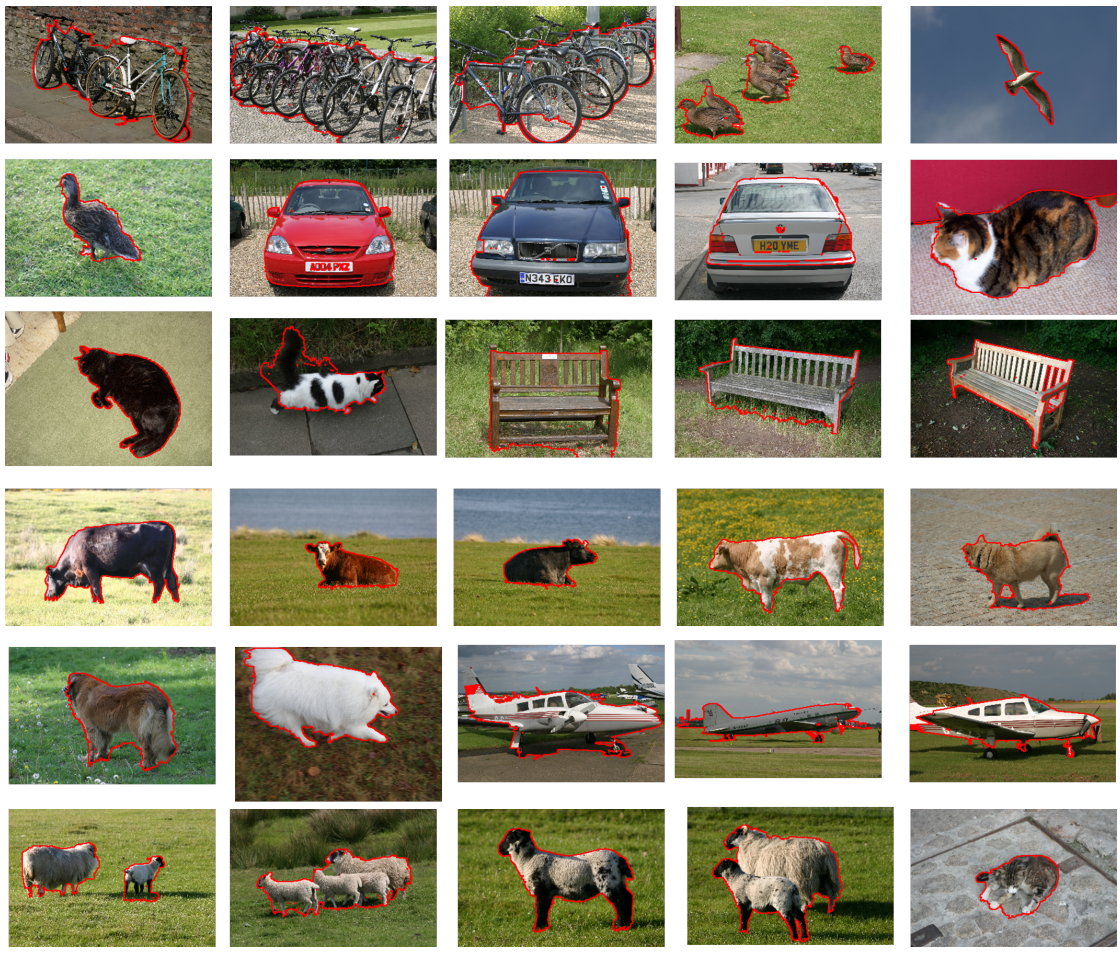

Figure 6.9: Qualitative results of our method on MSRC.

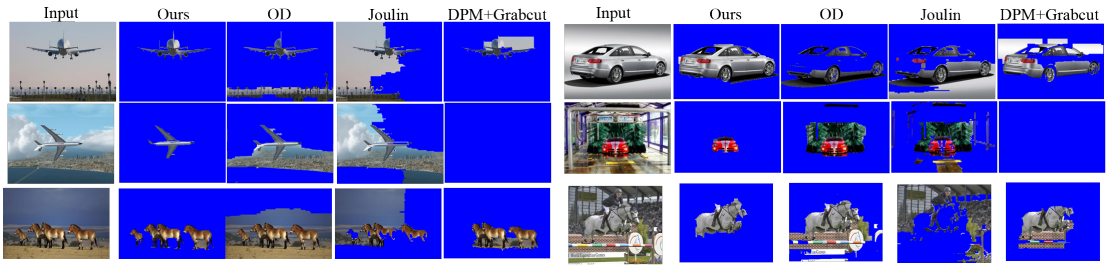

Figure 6.10: Comparison on the Object Discovery (OD) dataset of our method, OD, Joulin et al. [4], and DPM+Grabcut. Note how our method is able to segment non-salient objects while OD picks other areas apart from the object. DPM is unable to detect some objects.

comparison is fair. We also compared against [1, 2, 4].

The quantitative results can be found in Table 6.2. Since the images contain objects which are not salient in the image (more realistic images), our approach performs better than Object Discovery. It performs better than detector-based segmentation DPM+Grabcut, which has an expensive training phase and is not practical in our scenario. Also note that for airplanes our approach even performs better than the detector-based method upper bound. This is evidence of the high quality of our location prior as the initial GMM foreground and background color models derived from the location prior lead to better results than initializing color models from the correct tight bounding box input. Qualitative comparisons can be found in Figures 6.10 and 6.11. Fig. 6.12 shows more qualitative results on this dataset.

6.6.3 Results on Imagenet Dog

In order to test on a difficult real-world dataset where the object of interest is often small and not salient, we collected 100 images from ImageNet containing dogs. We show segmentation results on this dataset in Table 6.2. PASCAL VOC 2010 dog training pos-

Figure 6.11: More comparisons on the Object Discovery (OD) dataset of our method.

Figure 6.12: More results of our method on the Object Discovery (OD) dataset.

itives were used for training DPM+Grabcut. Qualitative results can be found in Figures 6.13 and 6.14.

6.7 Conclusion

In this paper we have proposed a new system for object selection. Our system has a far simpler interface for object selection, taking only the object name as input. In order to solve this problem we propose a exemplar-based localization method which relies on object retrieval. We break the image into object proposals and validate the presence of the object in the proposal. Location priors obtained in this way are then used to get an image specific appearance model and both are used to solve the segmentation problem in an MRF framework. We have introduced our own imageNet dog dataset and we outperform the state-of-the-art on a number of other datasets.

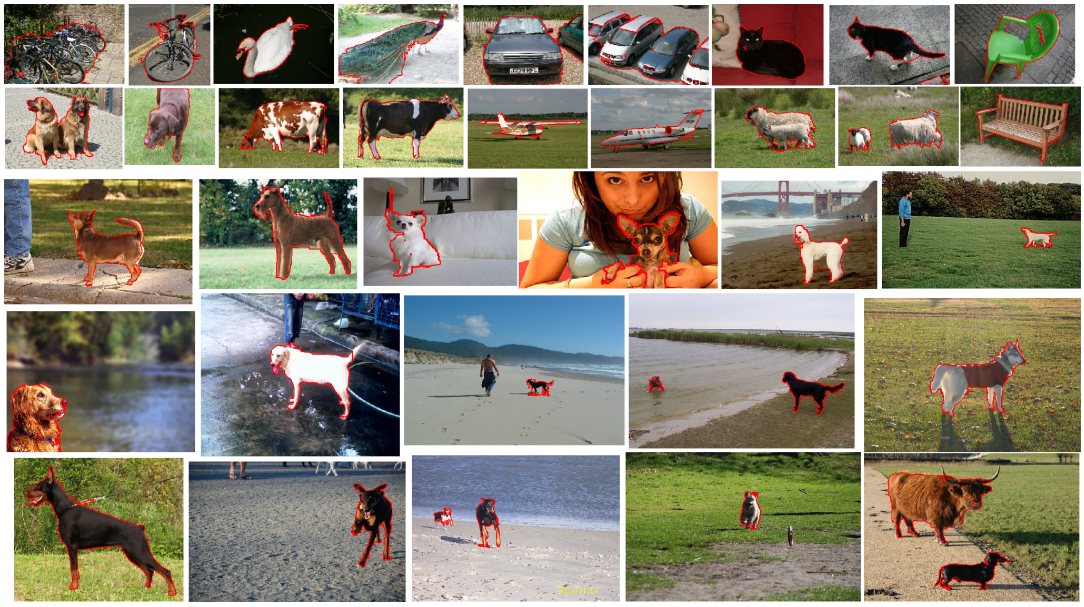

Figure 6.13: Qualitative results on MSRC (first two rows) and ImageNet-dog (last three rows).

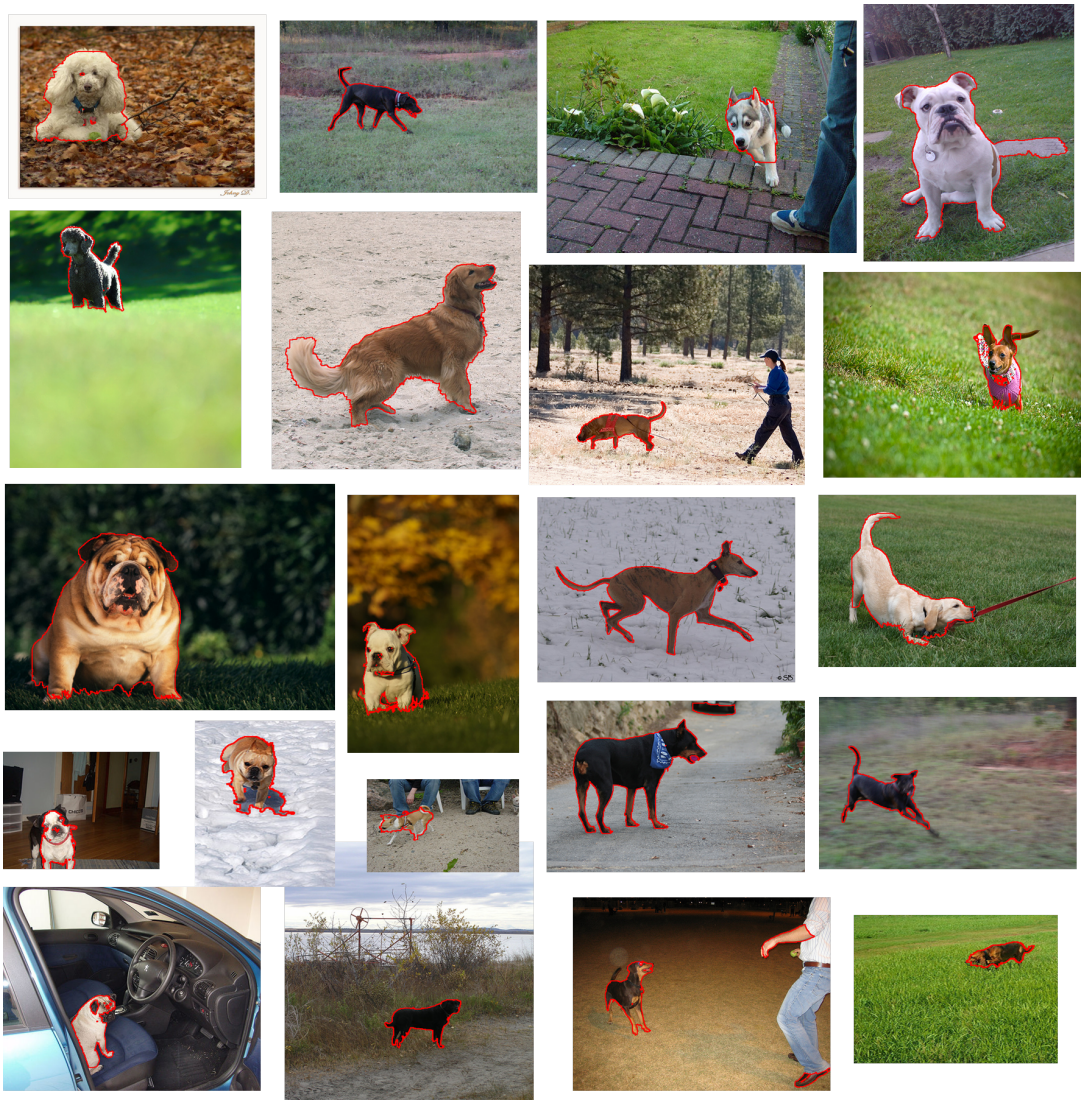

Figure 6.14: More qualitative results on ImageNet-dog.

Chapter 7: Conclusion

This dissertation explored the problem of understanding the visual world by reasoning about the objects present in it, providing contributions to the following tasks: efficient selection of filters for object detection (chapter 2 and 3), linear dimensionality reduction method composite discriminant factor (CDF) analysis (chapter 4), deep learning architecture for person re-identification (chapter 5), and semantic object selection for image editing (chapter 6).

In chapters 2, 3 and 4 we focus on providing efficient solutions for object detection. We start by showing that humans can help build better detectors as they can differentiate between a good filter and a bad one by visualizing them. This is shown by building an interactive framework for poselet selection. Our method significantly improves the training time, it takes about 5-10mins for a user to interactively select 100 poselets. As compared to 15Hrs (76% of 20Hrs) for computer to select them. We then extend the idea of filter selection and present an automatic and efficient method to select diverse set of discriminative filters. As an alternative to the expensive explicit evaluation that is often the bottleneck in many methods, this has the potential to dramatically alter the trade-off between accuracy of a part based model and the cost of training. In our experiments, we show that combined with LDA-HOG, an efficient alternative to SVM, for training the part

candidates, we can reduce the training time of a poselet model by an order of magnitude, while actually improving its detection accuracy. Moreover, we show that our approach to prediction of filter quality transcends specific detection architecture: rankers trained for poselets allow efficient filter/exemplar ranking for exemplar SVMs as well. Our work suggests that it is possible to evaluate the discriminative quality of a set of filters based purely on their intrinsic properties. Beyond direct savings in training time for part-based models, this evaluation may lead to speeding up part-based detection methods at *test time*, when used as an attention mechanism to reduce number of convolutions and/or hashing lookups. We then presented a linear dimensionality reduction method, Composite Discriminant Factor (CDF) analysis. CDF yields surprisingly good performance compared to PLS and SVM, and yields much more compact subspaces than PLS, leading to improved speed at runtime. The improvement is especially noticeable in the vehicle and human detection tasks, as well as on the multiclass action recognition task, suggesting that CDF is a good alternative to linear SVM for many state-of-the-art vision approaches.

In chapter 5 we have presented a novel deep architecture for person re-identification. We have proposed a novel architecture for finding relationships between two views, by designing cross-input neighborhood differences layer and a subsequent layer that summarizes these differences. We demonstrate the effectiveness of our method by performing a comprehensive evaluation of our approach on various data sets. On the large CUHK03 data set, our method outperforms the state-of-the-art by a huge margin. On the smaller CUHK01 data set (100 test IDs setting), whereas other deep methods overfit, our method is able to generalize and produce state-of-the-art results. We also show that models learned by our method on a large data set can be adapted to new, smaller data sets. We

demonstrate this by evaluating our method on two small data sets. On CUHK01 (486 test ids setting), we outperform all previous methods, and on VIPeR, our results are comparable to the state-of-the-art.

In the final part of the dissertation, we have presented a new system for object selection. We have presented a system which has far simpler interface for object selection, taking only the name of the object as input. In order to solve this problem we have proposed an exemplar-based localization method which relies on object retrieval. We have introduced our own imageNet dog dataset and we outperform the state-of-the-art on a number of other datasets. With the current proliferation of natural language interfaces for all sorts of tasks (e.g. Siri), we think our method will be very important for advancing image editing via natural language input.

Future directions for the work described in this dissertation include:

- Investigation of the role of class affinity in generalization of part quality; e.g., one might benefit from using part ranking from vehicle classes when the test class is also a vehicle.

- The idea of filter selection can be extended for convolution neural networks. There is a lot of redundancy in CNN filters which can be reduced if diverse set of filters are selected. For e.g., number of first layer filters in Krizhevsky et al. [83] can be reduced from 96 to 48.

- The use of additional composite candidates for CDF search (e.g., SVM), other subsequent classifiers, and extension to a kernel method for applications where kernel methods are practical.

- For person re-identification, different models can be trained for different parts of the body, and the scores from different part pairs can then be accumulated to reach a final decision section 5.8.1. Such a system may be helpful in handling severe occlusions and to identify people in images that have been taken across time (e.g., sitting in one view and standing in the other).

- Leveraging advancement made in the deep learning based classification literature for object selection. CNN trained on imagenet 1000 categories classification task can be used to select an object in an image if the target object is one of the 1000 classes. Concept of related classes and class affinity can be used in a zero shot setting when the target class is not present in 1000 classes.

- In order to improve image editing experience, we should go beyond object selection and exploit semantics from user queries. For example user should be able to issue more complex queries like, "make the dog next to the tree brighter" or "select the brown dog" to disambiguate among multiple dogs.

This dissertation makes an effort towards better understanding of the visual world by reasoning about the objects present in it. This is a challenging task, and to make progress we have to make advances on several fronts. We need better representations of visual categories that can enable accurate reasoning about their properties, as well as machine learning methods that can leverage big-data to learn such representations. For real time systems we need to make current frameworks more efficient without losing on performance. Successful methods for doing so can tremendously improve human interaction with machines for navigating large amounts of visual data. This is an exciting direction

of research with implications for building better systems for various vision tasks, as well as advancing AI in general.

Bibliography

[1] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," *CVPR*, June 2013. viii, xiii, 99, 101, 112, 113, 114, 115, 117

[2] A. Joulin, F. Bach, and J. Ponce, "Multi-class cosegmentation," in *CVPR*, 2012. viii, xiii, 101, 112, 113, 115, 117

[3] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade, "Distributed cosegmentation via submodular optimization on anisotropic diffusion," *ICCV*, pp. 169–176, 2011. viii, 101, 112, 113

[4] A. Joulin, F. R. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *CVPR*, 2010. viii, xiii, xiv, 101, 112, 113, 115, 117

[5] L. Mukherjee, V. Singh, and J. Peng, "Scale invariant cosegmentation for image groups," *CVPR*, pp. 1881–1888, 2011. viii, 101, 112, 113

[6] X. Shen, Z. Lin, J. Brandt, S. Avidan, and Y. Wu, "Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking," in *CVPR*, 2012. xiii, 103, 104, 105, 106

[7] E. Ahmed, S. Maji, G. Shakhnarovich, and L. S. Davis, "Using human knowledge to judge part goodness: Interactive part selection," in *CVPR Workshop on Computer Vision and Human Computation*, 2014. 8

[8] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *International Conference on Computer Vision*, 2009. 9, 15, 16, 19

[9] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *European Conference on Computer Vision*, 2010. 9, 10, 11, 15, 16, 18, 19

[10] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *International Conference on Computer Vision*, 2011. 9, 15, 16, 19, 21, 22

[11] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *European Conference on Computer Vision*, 2012. 9, 17

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition*, 2005. 9, 15, 18, 20

[13] T. Gao, M. Stark, and D. Koller, "What makes a good detector?–structured priors for learning from few examples," in *European Conference on Computer Vision*, 2012. 10, 17, 18, 27, 30

[14] E. Ahmed, G. Shakhnarovich, and S. Maji, "Knowing a good HOG filter when you see it: Efficient selection of filters for detection," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, 2014, pp. 80–94. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10590-1_6 14

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE PAMI*, 2010. 15, 17

[16] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *Computer Vision and Pattern Recognition*, 2013. 16

[17] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik, "Using k-poselets for detecting people and localizing their keypoints," in *Computer Vision and Pattern Recognition (CVPR)*, 2014. 16

[18] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich, "Viewpoint-aware object detection and pose estimation," in *International Conference on Computer Vision*, 2011. 17

[19] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *Computer Vision and Pattern Recognition*, 2009. 17

[20] Y. Aytar and A. Zisserman, "Immediate, scalable object category detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 17

[21] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004. 17

[22] S. Maji and J. Malik, "Object detection using a max-margin hough transform," in *Computer Vision and Pattern Recognition*, 2009. 17

[23] S. Maji and G. Shakhnarovich, "Part discovery from partial correspondence," in *Computer Vision and Pattern Recognition*, 2013. 17

[24] M. Aubry, B. Russell, and J. Sivic, "Painting-to-3D model alignment via discriminative visual elements," *ACM Transactions on Graphics*, vol. 33, no. 2, 2014. 17

[25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, 2005. 24

[26] S. Sarawagi and R. Gupta, "Accurate max-margin training for structured output spaces," in *ICML*, 2008. 24

[27] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, 2007. 24

[28] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, 1978. 25

[29] B. Hariharan, J. Malik, and D. Ramanan, "Discriminative decorrelation for clustering and classification," in *European Conference on Computer Vision*, 2012. 27

[30] V. I. Morariu, E. Ahmed, V. Santhanam, D. Harwood, and L. S. Davis, "Composite discriminant factor analysis," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014. 44

[31] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *JMLR*, 2003. 45, 63

[32] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011. 45, 47

[33] A. M. Martinez, A. M. Mart'inez, and A. C. Kak, "Pca versus lda," *PAMI*, 2001. 45, 48

[34] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Comput.*, 2004. 45, 48

[35] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995. 45

[36] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, 2000. 45

[37] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, 1998. 45

[38] S. Akaho, "A kernel method for canonical correlation analysis," *CoRR*, 2006. 45

[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005. 45, 47, 48, 57

[40] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," in *PAMI*, 2010. 45, 57, 58

[41] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, 2006. 45, 48, 60

[42] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006. 45

[43] S. Sadanand and J. Corso, "Action bank: A high-level representation of activity in video," in *CVPR*, 2012. 45, 48, 56, 57

[44] H. Wold, "Partial least squares," *Encyclopedia of Statistical Sciences*, 1985. 46, 48, 49, 50

[45] R. Rosipal and N. Krämer, "Overview and recent advances in partial least squares," in *SLSFS*, 2005. 46, 49

[46] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis, "Human Detection Using Partial Least Squares Analysis," in *ICCV*, 2009. 46, 48, 60

[47] A. Kembhavi, D. Harwood, and L. S. Davis, "Vehicle detection using partial least squares," *PAMI*, 2011. 46, 48

[48] http://crcv.ucf.edu/data/UCF50.php. 47, 55

[49] A. Asuncion and D. Newman, "UCI machine learning repository," http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html 47, 63

[50] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, 1991. 47

[51] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *PAMI*, 1997. 47

[52] B. Hariharan, J. Malik, and D. Ramanan, "Discriminative decorrelation for clustering and classification," in *ECCV*, 2012. 47, 58

[53] T.-K. Kim and R. Cipolla, "Canonical correlation analysis of video volume tensors for action categorization and detection," *PAMI*, 2009. 47

[54] X. Wang, T. X. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *ICCV*, 2009. 48

[55] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011. 48

[56] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *CVPR*, 2008. 48

[57] M. Pedersoli, J. Gonzàlez, A. D. Bagdanov, and J. J. Villanueva, "Recursive coarse-to-fine localization for fast object detection," in *ECCV*, 2010. 48

[58] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *ECCV*, 2010. 48

[59] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *CVPR*, 2011. 48

[60] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR*, 2011. 48

[61] U. G. Indahl, K. H. Liland, and T. Næs, "Canonical partial least squares-a unified pls approach to classification and regression problems," *Journal of Chemometrics*, 2009. 49

[62] R. Rosipal, P. P. Be, L. J. Trejo, N. Cristianini, J. Shawe-taylor, and B. Williamson, "Kernel partial least squares regression in reproducing kernel hilbert space," *JMLR*, 2001. 49

[63] K. J. Worsley, "An overview and some new developments in the statistical analysis of pet and fmri data," *Human Brain Mapping*, 1997. 55

[64] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *PAMI*, 2012. 57, 62

[65] R. Benenson, M. Mathias, T. Tuytelaars, and L. V. Gool, "Seeking the strongest rigid detector," in *CVPR*, 2013. 57

[66] J. J. Lim, C. L. Zitnick, and P. Dollar, "Sketch tokens: A learned mid-level representation for contour and object detection," in *CVPR*, 2013. 57

[67] G. Chen, Y. Ding, J. Xiao, and T. X. Han, "Detection evolution with multi-order contextual co-occurrence," in *CVPR*, 2013. 57

[68] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 68

[69] Y. Yang, J. Yang, J. Yan, S. Liao, D. Yi, and S. Li, "Salient color names for person re-identification," in *ECCV*, 2014. 70, 71

[70] Z. Zhang, Y. Chen, and V. Saligrama, "A novel visual word co-occurrence model for person re-identification," in *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 70, 71, 88

[71] R. Zhao, W. Ouyang, and X. Wang, "Learning mid-level filters for person re-identification," in *CVPR*, 2014. 70, 88, 90

[72] M. Koestinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *CVPR*, 2012. 70, 71, 86

[73] W. Li and X. Wang, "Locally aligned feature transforms across views," in *CVPR*, 2013. 70, 71

[74] N. Martinel, C. Micheloni, and G. Feresti, "Saliency weighted features for person re-identification," in *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 70, 71

[75] Z. Li, S. Chang, F. Liang, T. Huang, L. Cao, and J. Smith, "Learning locally-adaptive decision functions for person verification," in *CVPR*, 2013. 70, 71, 90

[76] R. Zhao, W. Ouyang, and X. Wang, "Person re-identification by salience matching," in *ICCV*, 2013. 70, 71, 88

[77] F. Xiong, M. Gou, O. Camps, and M. Sznaier, "Person re-identification using kernel-based metric learning methods," in *ECCV*, 2014. 70, 71

[78] S. Khamis, C. Kuo, V. Singh, V. Shet, and L. Davis, "Joint learning for attribute-consistent person re-identification," in *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 70, 71

[79] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014. 70, 71, 72, 81, 83, 85, 86, 87, 88, 91

[80] D. Yi, Z. Lei, and S. Z. Li, "Deep metric learning for practical person re-identification," *ICPR*, 2014. 71, 72, 81, 90

[81] W. Li, R. Zhao, and X. Wang, "Human re-identification with transferred metric learning," in *ACCV*, 2012. 72, 85, 88

[82] L. Bottou, "Stochastic gradient tricks," in *Neural Networks, Tricks of the Trade, Reloaded*, ser. Lecture Notes in Computer Science (LNCS 7700), G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 2012, pp. 430–445. [Online]. Available: http://leon.bottou.org/papers/bottou-tricks-2012 81

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf 83, 124

[84] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014. 84

[85] D. Gray, S. Brennan, and H. Tao, "Evaluating appearance models for recognition, reacquisition, and tracking," in *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007. 85

[86] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2009.167 86

[87] R. Zhao, W. Ouyang, and X. Wang, "Unsupervised salience learning for person re-identification," in *CVPR*, 2013. 86, 87, 88

[88] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, "Person re-identification by symmetry-driven accumulation of local features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 2360 –2367. 86, 88

[89] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 209–216. [Online]. Available: http://doi.acm.org/10.1145/1273496.1273523 86, 88

[90] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? metric learning approaches for face identification," in *ICCV*, Kyoto, Japan, Sep. 2009. [Online]. Available: http://hal.inria.fr/inria-00439290 87

[91] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009. 87, 88

[92] B. Mcfee and G. Lanckriet, "Metric learning to rank," in *In Proceedings of the 27th annual International Conference on Machine Learning (ICML)*, 2010. 87

[93] S. Pedagadi, J. Orwell, S. Velastin, and B. Boghossian, "Local fisher discriminant analysis for pedestrian re-identification," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, pp. 3318–3325, 2013. 89

[94] A. Mignon and F. Jurie, "Pcca: A new approach for distance learning from sparse pairwise constraints." in *CVPR*. IEEE, 2012, pp. 2666–2672. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2012.html#MignonJ12 89

[95] C. Liu, S. Gong, C. C. Loy, and X. Lin, "Person re-identification: What features are important?" in *ECCV Workshops (1)*, ser. Lecture Notes in Computer Science, A. Fusiello, V. Murino, and R. Cucchiara,

Eds., vol. 7583. Springer, 2012, pp. 391–401. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv2012w1.html#LiuGLL12 89

[96] W.-S. Zheng, S. Gong, and T. Xiang, "Person re-identification by probabilistic relative distance comparison," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 649–656. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2011.5995598 89

[97] B. Ma, Y. Su, and F. Jurie, "Bicov: a novel image representation for person re-identification and face verification," in *BMVC'12*, 2012, pp. 1–11. 89

[98] S. Bak, E. Corvee, F. Bremond, and M. Thonnat, "Multiple-shot human re-identification by mean riemannian covariance grid," in *Proceedings of the 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance*, ser. AVSS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 179–184. [Online]. Available: http://dx.doi.org/10.1109/AVSS.2011.6027316 89

[99] L. Bazzani, M. Cristani, A. Perina, and V. Murino, "Multiple-shot person re-identification by chromatic and epitomic analyses," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 898–903, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2011.11.016 89

[100] N. Gheissari, T. B. Sebastian, and R. Hartley, "Person reidentification using spatiotemporal appearance," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 1528–1535. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2006.223 89

[101] E. Ahmed, S. Cohen, and B. Price, "Semantic object selection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 96

[102] "Adobe system incorp. 2002. adobe photoshop user guide." 97

[103] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition," in *SIGGRAPH*, 1995, pp. 191–198. 97

[104] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *ICCV*, vol. 1, 2001, pp. 105–112. 97

[105] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," in *SIGGRAPH*, vol. 23, no. 3, 2004, pp. 309–314. 97, 100, 101, 109, 111, 114

[106] D. Batra, C. M. Univerity, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," in *CVPR*, 2010. 97

[107] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, "A bayesian approach to digital matting," in *CVPR*, 2001. 97

[108] A. Levin, D. Lischinski, and Y. Weiss, "A closed form solution to natural image matting," in *CVPR*, 2006. 97

[109] J. Wang and M. Cohen, "Optimized color sampling for robust matting," in *CVPR*, 2007. 97

[110] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing via label transfer," *PAMI*, vol. 33, no. 12, pp. 2368–2382, 2011. 97, 100

[111] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels," in *ECCV*, 2010. 97, 100

[112] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *CVPR*, 2013. 97, 100

[113] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection," in *CVPR*, 2011, pp. 409–416. 97, 100

[114] G. P. Laput, M. Dontcheva, G. Wilensky, W. Chang, A. Agarwala, J. Linder, and E. Adar, "Pixeltone: a multimodal interface for image editing," in *SIGCHI*, 2013. 97

[115] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *ECCV*, 2006. 99, 112

[116] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010. 101, 103, 105, 114

[117] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011. 101, 103, 105

[118] X. Shen, Z. Lin, J. Brandt, and Y. Wu, "Mobile product image search by automatic query object extraction," in *ECCV*, 2012. 101, 102

[119] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *ICCV*, 2011. 103, 104, 106

[120] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "Sift flow: Dense correspondence across different scenes," in *ECCV*, 2008, pp. 28–42. 103, 109

[121] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *NIPS 19*, 2007, pp. 545–552. 109

[122] D. Kuettel, M. Guillaumin, and V. Ferrari, "Segmentation propagation in imagenet," in *ECCV*, Oct. 2012. 109, 111

[123] D. Kuettel and V. Ferrari, "Figure-ground segmentation by transferring window masks," in *CVPR*, 2012. 109