University of California

Los Angeles

Structural Learning for Visual Inferences

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Quannan Li

2013

Abstract of the Dissertation

Structural Learning for Visual Inferences

by

Quannan Li

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Zhuowen Tu, Chair

In this work, we investigate the structural information in typical problems in both the machine learning and computer vision domains and propose effective yet efficient methods to tackle those problems. In particular, for structural labeling, we propose a fixed-point model which is able to learn/model long range structural contexts and is very efficient to train. For visual codebook learning, we propose a randomness and sparsity induced learning scheme which can fit to the local intrinsic structure of image patches. We also propose methods to tackle the problems of mid-level feature learning and object tracking where the structural information are helpful. For mid-level feature learning, we propose a fully automatic algorithm which harvests effective visual concepts from a large number of Internet images using text-based queries. For object tracking, we propose a disagreement-based approach which can be built on top of nearly any existing tracking systems by exploiting their disagreements. Our methods achieve state-of-the-art performance and high efficiency.

The dissertation of Quannan Li is approved.

To my grandmother, my parents, and my sisters

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Prof. Zhuowen Tu for his consistent instruction and support; Also, I would like to thank my committee members, Professor Song-chun Zhu, Professor Ying-nian Wu, and Professor Alan Yuille for their time to evaluate my work; Lastly, I would like to thank Professor Wenyu Liu, Xiang Bai, Jingdong Wang and my friends for their encouragement and help during my PhD's study.

Vita

2003–2007	B.S. (Electronic Engineering) Huazhong University of Science and Technology, China.
2007–2009	M.S. (Electronic Engineering) Huazhong University of Science and Technology, China.
2009–2013	Research Assistant Computer Science University of California, Los Angeles, California.

Publications

Quannan Li, Jiajun Wu and Zhuowen Tu, "Harvesting Mid-level Visual Concepts from Large-scale Internet Images", *CVPR* 2013

Quannan Li, Jingdong Wang, David Wipf and Zhuowen Tu, "Fixed-Point Model for Structured Labeling", *ICML* 2013

Quannan Li, Cong Yao, Liwei Wang and Zhuowen Tu, "Randomness and Sparsity Induced Codebook Learning with Application to Cancer Image Classification". *Workshop of Medical Computer Vision at CVPR*, 2012.

Quannan Li, Xinggang Wang, Wei Wang, Yuan Jiang, Zhi-Hua Zhou and Zhuowen Tu, "Disagreement-Based Multi-System Tracking". *Workshop of Detection and Tracking in Challenging Environments*, 2012

Yi Hong, Quannan Li, Jiayan Jiang and Zhuowen Tu. "Learning a Mixture of Sparse Distance Metrics for Classification and Dimensionality Reduction". *ICCV* 2011

CHAPTER 1

Introduction

The structural information has proven useful for tasks in computer vision and machine learning in that it can help to improve either the accuracy or the efficiency. When we talk about the structural information, the term *structure* can refer to three categories. The first category refers to the correlations among the entities in the structured outputs. For example, in the problem of Optical character recognition, the structured inputs are the scanned texts and the labels of the letters are correlated; in computer vision/image processing, the structured inputs are the pixels in an image, and the labels of the pixels are correlated. To model the contextual information of this category, there are already many classic methods proposed, e.g., [GG84, LMP01, TJH05, TGK03, TB10, HGS08]. These methods can model the contextual information well, but not very efficient to train. The second category refers to the structure or the intrinsic dimension of high-dimensional data spaces. Manifold learning methods such as Isomap [TSL00], LLE[RS00], Laplacian Eigenmaps [BN02] aim to find a low dimensional space to describe the original high dimensional space. In computer vision, as the feature data are often of high dimension, it is beneficial to investigate the local intrinsic dimensionality of the feature data space. The third category refers to the configuration of parts of the objects. It is usually simpler to model the parts of the objects and if we can model the configuration of the parts well, we can better recognize or track the objects. Typical methods of this type include [FGM10, FH05, SPY11].

In this work, we aim to model the three categories of structural information to tackle problems in computer vision and machine learning. For the contextual information of the first category, we propose a fixed-point model which is much more efficient to train than the classic models; for the second category, we propose a method to perform codebook learning by investigating the local regularities of subspaces (local data clusters) of the high dimensional feature space. In object recognition and object tracking, we have not headed to modeling the configurations of the parts of the object. Still, we propose a method to learn mid-level representation (visual concepts) for object recognition and a method to combine multiple tracking systems for robust object tracking. Figure 1.1 summarizes the relationship of the four methods of our work. Visual concept learning and disagreement-based multiple system tracking are dashed because we have not modeled the configurations of objects yet.

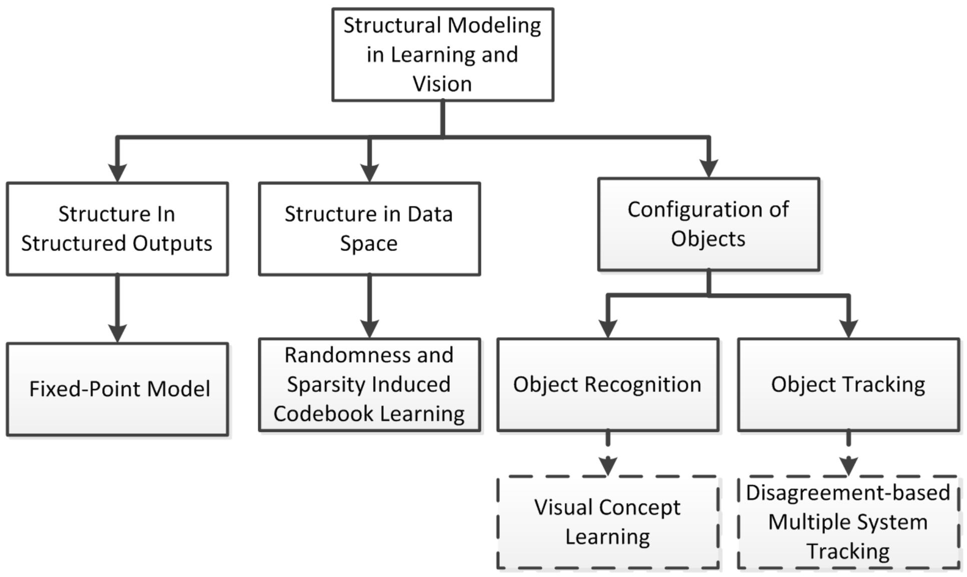

Structural Modeling in Learning and Vision
Structure In Structured Outputs
Structure in Data Space
Configuration of Objects
Fixed-Point Model
Randomness and Sparsity Induced Codebook Learning
Object Recognition
Object Tracking
Visual Concept Learning
Disagreement-based Multiple System Tracking

Figure 1.1: Relationship of the methods in our work.

1.1 Fixe-Point model for structural labeling

Structural labeling is to jointly assign labels to all nodes in the structured input as a joint output. This problem is very fundamental since structured inputs and outputs are common in a wide range of applications. It is also very difficult because of the correlations among the structured outputs.

Traditional methods such as [GG84, LMP01, TJH05, TGK03] can model the correlations of the structured labels but are limited to capturing a few neighborhood interactions due to the heavy computational burdens in their training and testing (inference) stages. Recently, deep layered models [TB10, HGS08, DLM09] are proposed to take the outputs of classifiers of the current layer as added features to classifiers of the next layer. The layered models [TB10, HGS08] are able to model complex and long range contexts more effectively but have to learn a series of classifiers.

In this work, we propose a simple but effective solution to the structured labeling problem: a fixed-point model. The fixed-point model is an algorithm with a new perspective on layered models; we aim to find a fixed-point function with the structured labeling being both the output and the input. The learned function captures rich contextual information and is easy to train and test. Our approach alleviates the burden in learning multiple/different classifiers in different layers. We devise a training strategy for our method and provide justifications for the fixed-point function to be a contraction mapping. On several widely used benchmark datasets, the proposed method observes significant improvement in both performance and efficiency over many state-of-the-art algorithms.

1.2 Randomness and Sparsity Induced Codebook Learning

Codebook learning deals with the fundamental representation problem which is one of the central research topics in computer vision. In codebook learning, the data are usually high-dimensional and live in complex manifolds. With their intrinsic and mathematical properties gradually unfolded, research in three general directions has led to significant progress on classification, recognition, and compression: (1) ensemble learning, (2) divide-and-conquer, and (3) sparse coding. Ensemble learning approaches [BRE96, FS97, BRE01] have shown to be among the best choices for classifiers [CN06, CKY08] with their superior robustness stemming from the voting of multiple independent or complementary experts (weak learners). The divide-and-conquer strategies [BEN80, QUI86, CZL07, CZL07] divide the data space into subspaces which are often easier to deal with and are more appropriate for high-dimensional data spaces. Sparse representations such as compressed sensing [CT05] and LASSO [TIB96] have gained a great deal of success and popularity since high-dimensional data within intrinsic lower dimension can be well represented by sparse samples of high dimension.

In this work,we propose a new codebook learning algorithm, Randomized Forest Sparse Coding (RFSC), by harvesting the three concepts. Given a set of training data, a randomized tree can be used to perform data partition (divide-and-conquer); after a tree is built, a number of bases are learned from the data within each leaf node for a sparse representation (subspace learning via sparse coding); multiple trees with diversities are trained (ensemble), and the collection of bases of these trees constitute the codebook. These three concepts in our codebook learning algorithm have the same target but with different emphasis: subspace learning via sparse coding makes a compact representation, and reduces the information loss; the divide-and- conquer process efficiently obtains the local data clusters; an ensemble of diverse trees provides additional robustness. We have conducted classification experiments on cancer images as well as a variety of natural image datasets and the experiment results demonstrate the efficiency and effectiveness of the proposed method.

1.3 Harvesting Mid-level Visual Concepts from Large-scale Internet Images

The inventions of robust and informative low-level features such as SIFT [LOW04], HOG [DT05], and LBP [OPH96] have been considered as one of the main advances/causes for the recent success in computer vision. Beyond low-level features, obtaining effective mid-level representations has become increasingly important and there have been many recent efforts made along the line of attribute learning [FEH09, PG11a, LSX10]. These approaches, however, are mostly focused on supervised or active learning where a considerable amount of human efforts are required to provide detailed manual annotations. The limitations to the previous supervised attribute learning methods are thus three-fold: (1) accurate data labeling is labor-intensive to obtain, (2) the definition of attributes is often intrinsically ambiguous, (3) the number of attributes and training images are hard to scale.

In this work, we propose a fully automatic algorithm which harvests visual concepts from a large number of Internet images (more than a quarter of a million) using text-based queries. We take the advantage of having massive well-organized Google and Bing image data; visual concepts (around $14,000$) are automatically exploited from images using word-based queries; We apply the multiple instance learning formulation [ATH02] to exploit common patterns from retrieved images, which have a high degree of relevance to the query words; We use bottom-up saliency detection to reduce the search space by finding potential candidates. Using the learned visual concepts, we show state-of-the-art performances on a variety of benchmark datasets. The good performance demonstrates the effectiveness of the learned mid-level representations: being able to generalize well to different image sets.

1.4 Disagreement-Based Multi-System Tracking

Object tracking has been a long standing problem in vision. Once a tracker gets initialized, it starts to track the target in a video by making a prediction about the location of the target and updating its object model (location, appearance, and shape) based on the prediction. With the recent success in detection-based tracking approaches, an increasing amount of work has treated the tracking problem as a semi-supervised learning problem [AVI05, TBZ07, GLB08, LRL08, BYB09]. Due to the errors introduced in both the prediction and model updating stage, nearly any tracker will eventually fail with the errors being accumulated over the time.

In this work, we tackle the tracking problem from a fusion angle and propose a disagreement-based approach. While most existing fusion-based tracking algorithms work on different features or parts, our approach can be built on top of nearly any existing tracking systems by exploiting their disagreements. In contrast to assuming multi-view features or different training samples, we utilize existing well-developed tracking algorithms, which themselves demonstrate intrinsic variations due to their design differences. Our intuition is to find the location where the current tracker is confident but disagrees with other trackers, while other trackers reach a high degree of agreement by seeking a balance between the current tracker and the level of agreements among other trackers. We present encouraging experimental results as well as theoretical justification of our approach. On a set of benchmark videos, large improvements (20% $\sim$ 40%) over the state-of-the- art techniques have been observed.

CHAPTER 2

Fixe-Point model for structural labeling

2.1 Introduction

Here we study the problem of predicting a labeling for a structured input, which is denoted as a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ corresponds to a data entry with its features denoted as $\mathbf{x}_i$; the objective of the structured labeling task is to jointly assign labels $\mathbf{y} = (y_i : i = 1..|\mathcal{V}|)$ ($y_i \in \mathcal{L}$, $\mathcal{L}$ is the label space) to all nodes $\mathcal{V} = (v_i : i = 1..|\mathcal{V}|)$ as a joint output. This problem is fundamental since structured inputs and outputs are common in a wide range of applications. For example, in computer vision/image processing, a structured input is an image of all pixels, and the structured outputs are the corresponding labels of these pixels. There are correlations among the structured outputs, denoted by the edges between the nodes, and the correlation may occur between neighboring nodes, or the nodes relatively distant apart. These correlations make the structured prediction problem a difficult task.

A simple scheme is to treat the structured outputs as independent entries and apply the standard classification/regression algorithms. Such a scheme is straightforward, but it loses the important interdependency information, which is crucial in modeling and understanding the structured data. The other extreme of the solution is to treat each instance of $\mathbf{y} = (y_1, .., y_{|\mathcal{V}|})$ as a single label and transform the problem into a multi-class classification problem. This implementation is infeasible because the space of the output labels is exponentially large at the size of $|\mathcal{L}|^{|\mathcal{V}|}$.

Markov random fields (MRF) [GG84] and conditional random fields (CRF) [LMP01] have been widely used to model the correlations of the structured labels. However, due to the heavy computational burdens in their training and testing (inference) stages, MRF and CRF are often limited to capturing a few neighborhood interactions, and thus, limiting their modeling capabilities. Structural SVM methods [TJH05] and maximum margin Markov networks (M^3N) [TGK03] model the correlation in a similar way as the CRF, but they try to specifically maximize the prediction margin. These approaches are also limited in the range of contexts due to the high computational demand. When long range contexts are used, approximations should be used to trade-off the accuracy and the running time [FJ08].

Recently, layered models [TB10, HGS08, DLM09], in the spirit of stacking [WOL92], are proposed to take the outputs of classifiers of the current layer as added features to classifiers of the next layer. Since these approaches perform direct label prediction as functions instead of performing inferences as in MRF or CRF, the layered models [TB10, HGS08] are able to model complex and long range contexts.

In this paper, we look into the structured labeling problem from a different angle and develop a simple yet effective approach, a fixed-point model. We introduce a contextual prediction function $f : (\mathbf{x}, \mathcal{L}^{|\mathcal{V}|-1}) \rightarrow \mathcal{L}$ with the output being the labeling of an individual node and the input being both its features and the labeling of the rest of the nodes (or its neighbors). The overall fixed-point function $\mathbf{f} : (\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}, \mathcal{L}^{|\mathcal{V}|}) \rightarrow \mathcal{L}^{|\mathcal{V}|}$ is a vector form of the contextual prediction function of the nodes, and is trained with the property of a contraction mapping so that an iterative solution is applicable in the prediction process. We also analyze conditions for ensuring that our training strategy leads to a contraction mapping, provably so in certain cases. Not only does the learned fixed-point function preserve the modeling capability of the layered models [TB10, HGS08], but also it is simper and much easier to scale since it only consists of a single layer function.

2.2 Related Work

The conditional random fields (CRF) model [LMP01] is a state-of-the-art work for solving the structured prediction problem. In the max-margin Markov networks [TGK03, TJH05], the authors propose to maximize the margin for structured output, in a spirit similar to the multi-class SVM method [WW98]. Due to the computational demand in both the training and the testing stages, usually only a small number of interactions among the neighboring outputs are included in both the CRF and the M^3N. The hidden Markov models [RAB89] share a similar property in modeling the graph connections.

The layered contextual models [TB10, HGS08] train a sequences of classifiers using the output of the previous layers as additional features to the next layer; on tasks where the long range contexts play a significant role, e.g., the OCR task, they greatly outperform the CRF and M^3N (as shown in the experiments). The proposed fixed-point model has a similar modeling capability to model long range contexts as the layered models. During the training process, while the layered models train a series of classifiers, the fixed-point model trains a single classification/regression function which assumes a stable status for the ground-truth labeling. Layered models have to compute the classification scores for each training sample as the input to the next layer, thus limiting their capability to scale up. On the contrary, the fixed-point model is much faster to train than the layered models, and thus is much more scalable. In addition, the convergence behavior of the layered models has not been clearly stated so far, whereas the proposed fixed-point model provides a contraction mapping interpretation to the convergence.

The pseudo-likelihood algorithm in [JUL75] models the conditional probability of an entry based on its neighborhoods; in [SMJ10], a pseudo-max framework is introduced to approximate the exponential number of constraints by a polynomial number of constraints; in structured output-associative regression (SOAR) [BS09], the output components, other than the one being considered, are used as auxiliary features to train a vector of regression functions for the task of image reconstructions and human pose estimation. Compared with SOAR, the main purpose of the proposed fixed-point model is to train a fixed-point function that assumes the stable status of the structured labels; in SOAR, a generalized linear regression function is trained for reconstruction, whereas we study the structured labeling problem by exploring rich contextual correlations; the lack of analysis in the learned function also leaves the convergence in SOAR untouched; on the contrary, our algorithm is not limited to generalized linear regressions and, existing methods such as logistic regression and random forest [BRE01] can be used too.

There are also other related algorithms. In [COL02], an averaged perceptron algorithm is proposed: the training samples are processed iteratively; once a mistake occurs, the weight vector is updated according to the prediction error and the final weight vector is a weighted version of all the vectors that have appeared. In [NG07], an ensemble method is proposed to transform the predictions of different models into a chain with state transition matrices and then dynamic programming is used to get the voted prediction result. The underlying mechanism of the fixed-point model is different from that of these algorithms, and we will compare the performance in the experimental section.

It is worth mentioning that the proposed fixed-point model is not a method merely designed to balance the performance and learning-time; it provides a new way of thinking about the structured learning problem by investigating a shallow model (instead of cascaded approaches with deep layers) and also having the capability to incorporate rich structural/contextual information with effective and efficient inference.

2.3 The Fixed-Point Model

2.3.1 Model Description

In this paper, we are interested in the structured labeling task for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The edges $\mathcal{E}$ decides the graph topology, and hence the neighborhoods of each node. For instance, in sequence labeling, where the nodes $\{v_1, v_2, \cdots, v_n\}$ in $\mathcal{V}$ form a chain, we can specify the neighborhood $\mathcal{N}_i$ of v_i to be the m nodes preceding and after it, i.e., $\mathcal{N}_i = \{v_{i-m/2}, v_{i-m/2+1}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+m/2}\}$. We use m to denote the number of neighbors a node can have in the neighborhood specification.

We assume that our problem is a binary-classification problem where $\mathcal{L} = \{-1, +1\}$. To this end, we train a contextual prediction function which outputs the labeling of the node. Note that, as a lexical category label, y_i cannot be used in the equation/function directly. Instead, we can represent y with a labeling confidence q. For the binary class case, if $y_i = 1$, $q_i = 1$ and if $y_i = -1$, $q_i = 0$. At the prediction process, the label y_i is unknown, and thus q can be relaxed to a real value ranging in $[0, 1]$. We use $\mathbf{q}_{\mathcal{N}_i}$ to denote the labeling of the neighborhood of v_i and use $\mathbf{q}$ to denote the labeling of all the nodes in $\mathcal{G}$. This can easily be extended to multiclass problems by encoding the labeling with a matrix.

For each node v_i, the contextual prediction function f takes in both v_i's feature $\mathbf{x}_i$ and the labeling $\mathbf{q}_{\mathcal{N}_i}$ of its neighborhood. The contextual prediction function f can be formulated as

$$q_i = f(\mathbf{x}_i, \mathbf{q}_{\mathcal{N}_i}; \boldsymbol{\theta}), \tag{2.1}$$

where f is a regression function within range $[0, 1]$, and $\boldsymbol{\theta}$ is the parameter of the function. From Equation 2.1, the labeling $\mathbf{q}$ of all the nodes can be written in a vector form,

$$\mathbf{q} = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta}), \tag{2.2}$$

where $\mathbf{q} = [q_1, q_2, \cdots, q_n]^T$, $\mathbf{f}(\cdot) = [f(\mathbf{x}_1, \mathbf{q}_{\mathcal{N}_1}; \boldsymbol{\theta}), f(\mathbf{x}_2, \mathbf{q}_{\mathcal{N}_2}; \boldsymbol{\theta}), \cdots, f(\mathbf{x}_n, \mathbf{q}_{\mathcal{N}_n}; \boldsymbol{\theta})]^T$.

As from Equation 2.2, the labeling $\mathbf{q}$ appears as both the output as well as part of the input. Given the labeling $\mathbf{q}$ and the features $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ of the training data, we learn the parameter $\boldsymbol{\theta}$.

To get the labeling of a structured input $\mathcal{G}$, one can solve for the non-linear equation set $\mathbf{q} = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta})$, which is generally a difficult task. In this paper, we focus on a type of functions $\mathbf{f}$ that assumes the property of a contraction mapping, i.e., having a stable status (an attractive fixed-point) for each structured input. When using the ground-truth labeling in the training process, the ground-truth labeling is assumed to be the stable status and the existence of the stable status leads to the fixed-point iteration in the prediction process: $\mathbf{q}^t = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}^{t-1}; \boldsymbol{\theta})$ and $\mathbf{q}^t \to \mathbf{q}$ as $t \to \infty$. We name the functions $\mathbf{f}$ with such a property *the fixed-point functions (models)*. In the following subsections, we provide a sufficient condition for the fixed-point model, propose the learning strategy, and describe the training and testing processes.

2.3.2 Contraction Condition for the Fixed-Point Model

In this section, we give a sufficient condition for $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta})$ to be the fixed-point model and illustrate it using a logistic regression model first.

Our derivation is based on the Banach Fixed-Point theorem [BAN22]: for a complete metric space (X, dist) and a mapping $\mathcal{F} : X \to X$, if there exists a non-negative real number $\rho < 1$ such that,

$$\mathrm{dist}(\mathcal{F}(a), \mathcal{F}(b)) \leq \rho \times \mathrm{dist}(a, b), \forall a, b \in X, \tag{2.3}$$

then $\mathcal{F}$ is a contraction mapping and it has a unique fixed-point.

Since for a node v_i, its feature $\mathbf{x}_i$ is given and fixed, we thus for notational simplicity we can simply write $q_i = f_i(\mathbf{q}_{\mathcal{N}_i}; \boldsymbol{\theta})$ and $\mathbf{q} = \mathbf{f}(\mathbf{q}; \boldsymbol{\theta})$.

Assuming f_i is a continuously differentiable real-valued function, then according to the mean value theorem for a scalar function of several variables, $\forall\ \mathbf{q}, \overline{\mathbf{q}}$

$$f_i(\mathbf{q}_{\mathcal{N}_i}; \boldsymbol{\theta}) - f_i(\overline{\mathbf{q}}_{\mathcal{N}_i}; \boldsymbol{\theta}) = \left\langle \nabla f_i(\tilde{\mathbf{q}}_{\mathcal{N}_i}; \boldsymbol{\theta}), (\mathbf{q}_{\mathcal{N}_i} - \overline{\mathbf{q}}_{\mathcal{N}_i}) \right\rangle, \tag{2.4}$$

where $\nabla f_i(\tilde{\mathbf{q}}_{\mathcal{N}_i}; \boldsymbol{\theta})$ is the gradient of f_i at $\tilde{\mathbf{q}}_{\mathcal{N}_i} = (1 - c_i)\mathbf{q}_{\mathcal{N}_i} + c_i\overline{\mathbf{q}}_{\mathcal{N}_i}$, for some $0 < c_i < 1$. For $\mathbf{f}$,

$$\mathbf{f}(\mathbf{q}; \boldsymbol{\theta}) - \mathbf{f}(\overline{\mathbf{q}}; \boldsymbol{\theta}) \;\; = \;\; J_{\mathbf{f}}(\mathbf{q} - \overline{\mathbf{q}}), \tag{2.5}$$

where $J_{\mathbf{f}}$ is a matrix of coordinate-wise derivatives, and its i-th row corresponds to $\nabla f_i(\tilde{\mathbf{q}}_{\mathcal{N}_i}; \boldsymbol{\theta})$: for the (i, k)-th element of $J_{\mathbf{f}}$, if the node v_k is a neighbor of node v_i, $J_{\mathbf{f}_{i,k}} = \frac{\partial f_i}{\partial q_k}|_{\tilde{q}_k}$ ($\tilde{q}_k = (1 - c_i)q_k + c_i\overline{q}_k$); if v_k is not in the neighborhood of v_i, $J_{\mathbf{f}_{i,k}} = 0$. Clearly

$$\begin{aligned} \frac{\|\mathbf{f}(\mathbf{q}; \boldsymbol{\theta}) - \mathbf{f}(\overline{\mathbf{q}}; \boldsymbol{\theta})\|}{\|\mathbf{q} - \overline{\mathbf{q}}\|} &= \frac{\|J_{\mathbf{f}}(\mathbf{q} - \overline{\mathbf{q}})\|}{\|\mathbf{q} - \overline{\mathbf{q}}\|} \\ &\leq \max_{\mathbf{q} - \overline{\mathbf{q}} \neq \mathbf{0}} \frac{\|J_{\mathbf{f}}(\mathbf{q} - \overline{\mathbf{q}})\|}{\|\mathbf{q} - \overline{\mathbf{q}}\|} = \|J_{\mathbf{f}}\|, \end{aligned} \tag{2.6}$$

where $\|J_{\mathbf{f}}\|$ denotes the matrix norm on $J_{\mathbf{f}}$ induced from some vector norm $\|\cdot\|$. For example, $\|J_{\mathbf{f}}\|_1 = \max_{1 \leq k \leq n} \sum_{i=1}^{n} \left| \frac{\partial f_i}{\partial q_k}|_{\tilde{q}_k} \right|$, the induced ℓ_1 norm of $J_{\mathbf{f}}$. We then have the following:

Lemma 1 *If* $\|J_{\mathbf{f}}\| < 1\ \forall\ \mathbf{q}, \overline{\mathbf{q}}$, *then* $\mathbf{f}$ *is a contraction mapping.*

2.3.2.1 Contraction for logistic regression

Now we assume a linear logistic regression model. $\boldsymbol{\theta}$ can be decomposed into $\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in \mathcal{R}^{d \times 1}$, d is the dimension of $\mathbf{x}_i$) and $\boldsymbol{\beta}$ ($\boldsymbol{\beta} \in \mathcal{R}^{m \times 1}$), corresponding to the appearance feature $\mathbf{x}_i$ and the contextual feature $\mathbf{q}_{\mathcal{N}_i}$ respectively. $q_i = f_i(\mathbf{q}_{\mathcal{N}_i}; \boldsymbol{\theta})$

can be formulated as

$$q_i = \frac{\exp(\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle + \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle)}{1 + \exp(\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle + \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle)}. \tag{2.7}$$

In the following, we use $I(k, j)$ to denote the index of the node that has v_k as its j-th neighbor and define an auxiliary function

$$h_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - (\sum_{j=1}^m \beta_j - \sum_{j=1}^m |\beta_j|)/2)}{(1 + \exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - (\sum_{j=1}^m \beta_j + \sum_{j=1}^m |\beta_j|)/2))^2}. \tag{2.8}$$

In Lemma 2, we give a sufficient condition for $\|J_{\mathbf{f}}\|_1 < 1$ for logistic regression.

Lemma 2 *For the logistic regression model, if* $\max_{1 \le k \le n} \sum_{j=1}^m |\beta_j| \, h_{I(k,j)}(\boldsymbol{\alpha}, \boldsymbol{\beta}) < 1$, $\|J_{\mathbf{f}}\|_1 < 1$ *and the fixed-point function* $\mathbf{f}$ *is a contraction mapping.*

Proof If v_k is the j-th neighbor of v_i, then the partial derivative

$$\frac{\partial f_i}{\partial q_k} = \frac{\beta_j \exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle)}{(1 + \exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle))^2} \tag{2.9}$$

. As q_k is in the range $[0, 1]$, for the term $\langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle$, its minimum is $(\sum_{j=1}^m \beta_j - \sum_{j=1}^m |\beta_j|)/2$, the sum of the negative entries of $\boldsymbol{\beta}$, and its maximum is $(\sum_{j=1}^m \beta_j + \sum_{j=1}^m |\beta_j|)/2$, the sum of the positive entries of $\boldsymbol{\beta}$. So $\frac{\exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle)}{(1+\exp(-\langle \boldsymbol{\alpha}, \mathbf{x}_i \rangle - \langle \boldsymbol{\beta}, \mathbf{q}_{\mathcal{N}_i} \rangle))^2} \le h_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $|J_{\mathbf{f}_{i,k}}| = \left| \frac{\partial f_i}{\partial g_k} |_{\tilde{q}_k} \right| \le |\beta_j| \, h_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Ignoring the boundary effect of the structured input, the k-th absolute column sum of $J_{\mathbf{f}}$ sums over the m nodes that have v_k as their neighbor giving

$$\sum_{i=1}^n |J_{\mathbf{f}_{i,k}}| = \sum_{i: v_k \in \mathcal{N}_i} \left| \frac{\partial f_i}{\partial q_k} |_{\tilde{q}_k} \right| \le \sum_{j=1}^m |\beta_j| \, h_{I(k,j)}(\boldsymbol{\alpha}, \boldsymbol{\beta}). \tag{2.10}$$

Thus, $\|J_{\mathbf{f}}\|_1 \le \max_{1 \le k \le n} \sum_{j=1}^m |\beta_j| \, h_{I(k,j)}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. If $\max_{1 \le k \le n} \sum_{j=1}^m |\beta_j| \, h_{I(k,j)}(\boldsymbol{\alpha}, \boldsymbol{\beta}) < 1$, $\mathbf{f}$ is then a contraction mapping. ∎

The value of $\mathbf{x}$ plays a role in the constraint requirement in Equation 2.10. So long as the value of $\mathbf{x}$ satisfies the constraint requirement in Equation 2.10, the fixed-points can be guaranteed. Another possible sufficient condition is $\|\boldsymbol{\beta}\|_1 < 1$ as $|J_{\mathbf{f}_{i,k}}| = |\beta_j| \, q_i(1 - q_i) < |\beta_j|$. This condition is much simpler but more difficult to satisfy because it ignores dependency on $\mathbf{x}$ entirely.

2.3.2.2 Contraction in general cases

The condition described above can be used as constraints for the fixed-point function in the training process when f is restricted to a logistic regression function. We now briefly discuss a scheme for learning a contraction function in a more general setting, which can be used to implicitly enforce the contraction condition for other functions.

It is well-known that adding some amount of input noise (also called input jitter) during the training process can improve the robustness of neural network classifiers [ROM92]. Moreover, this is generally true with recursive models such as the algorithm being proposed herein, in part by favoring the contraction condition. For example, when training the function $\mathbf{q} = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta})$, we may produce some replica $Q = \{\mathbf{q} + \boldsymbol{\delta}_r, r = 1..R\}$ by introducing small random perturbations $\boldsymbol{\delta}_r$. This small amount of randomness is added to the input $\mathbf{q}$ of the function f while keeping the targeted output the same as the ground-truth $\mathbf{q}$, leading to an augmented training set. Given modest assumptions on the space of classifiers and training algorithms, it is possible to show that when a sufficient number of such replica are included with suitable distribution, then a contraction mapping will be obtained as part of the learning process with high probability. Intuitively, this occurs because these replica will effectively reduce the relative importance of $\mathbf{q}$ as an input feature to $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta})$. In the case of logistic regression, this is tantamount to reducing the magnitude of the coefficients $\boldsymbol{\beta}$; however, with other classifiers the effect may be less transparent. While it is difficult to know the optimal distribution of such replica a priori, we have found empirically that a contraction mapping is consistently obtained without sensitivity to this distribution. For evaluation purposes, the gradient of any classifier can be computed in principle, numerically or analytically, to examine if the contraction condition is satisfied in a particular region. Additionally, if a classifier is ever observed to violate the contraction condition, we can always retrain after increasing the number and/or magnitude of the replica. Given some assumptions about the classifier and the replica, we next briefly discuss efficient methods for checking (at least locally around the training data) whether or not a contracting function has been obtained.

From Lemma 1, we want to guarantee that $\|J_{\mathbf{f}}\| < 1$, for some norm $\|\cdot\|$. If we choose the induced ℓ_∞ norm, this corresponds to the requirement that all rows of $J_{\mathbf{f}}$ have ℓ_1 norm less than one. This can be guaranteed if the gradient of each f_i with respect to $\mathbf{q}$ is less than one for all $\mathbf{q}$. Let $\psi_i(\mathbf{q})$ denote this gradient. Using a Taylor series expansion we can approximate each function f_i as

$$f_i(\mathbf{q}+\boldsymbol{\delta}_r) = f_i(\mathbf{q}) + \boldsymbol{\delta}_r \cdot \psi_i(\mathbf{q}) + O(\partial^n f_i, n \geq 2), \tag{2.11}$$

where for simplicity we use $f_i(\mathbf{q})$ to denote the i-th element of $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \mathbf{q}; \boldsymbol{\theta})$. It is not uncommon to assume a relatively smooth function f_i with small higher-order derivatives $O(\partial^n f_i), n \geq 2$. Moreover, we may also assume in some situations that the $\boldsymbol{\delta}_r$ are small, in which case higher-order terms can be largely ignored. Let $f_i(\mathbf{q}+\boldsymbol{\delta}_r) - f_i(\mathbf{q}) = e_r$ for all $r = 1..R$. Given the first-order Taylor-series approximation, we then have

$$\boldsymbol{\delta}_r \cdot \psi_i(\mathbf{q}) \approx e_r, \;\; r = 1..R. \tag{2.12}$$

The above constraints represent a linear system that can be viewed as random samples of the unknown $\psi_i(\mathbf{q})$. These samples, which can be efficiently collected and monitored during the training process, can then be used to help determine whether or not the contraction condition is satisfied (at least in the locality of the training data). Depending on the neighborhood structure of the graph, we know that $\psi_i(\mathbf{q})$ will typically be sparse, with nonzero-valued locations inferred from the edges. In cases where this degree of sparsity is sufficiently high relative to the number of replica, we can simply solve for $\psi_i(\mathbf{q})$ directly via the above linear system. However, when this is not possible, we may still potentially estimate whether $\|\psi_i(\mathbf{q})\|_1 < 1$.

For example, assume for simplicity that the replica $\boldsymbol{\delta}_r$ are iid Gaussian distributed with zero mean and known covariance $\sigma^2 I$ (other distributions can be accommodated as well). It then follows that each e_r represents an iid sample from a zero-mean Gaussian with variance $\sigma^2\|\psi_i(\mathbf{q})\|_2^2$. Given R such samples, it is a simple matter to design any number of standard statistical tests to infer the likelihood that $\|\psi_i(\mathbf{q})\|_2^2 < C$ for any constant C. So we need only determine some C sufficiently small such that we ensure the contraction condition holds, namely $\|\psi_i(\mathbf{q})\|_1 < 1$ with high probability. Now assume that the number of significant elements in $\psi_i(\mathbf{q})$ is less than or equal to some value τ (in addition to zero-valued elements enforced by the graph, there are typically many other elements with marginal influence, although these locations may not be known). Using the well-known relationships among p-norms, if $\tau\|\psi_i(\mathbf{q})\|_2^2 < 1$, then $\|\psi_i(\mathbf{q})\|_1 \leq \sqrt{\tau}\|\psi_i(\mathbf{q})\|_2 < 1$. So $C = 1/\tau$ is an appropriate choice.

This methodology can be loosely used to show that our learned function satisfies the contraction condition of $\|\psi_i(\mathbf{q})\|_1 < 1$ at the $\mathbf{q}$ from the training data and in neighboring regions such that an affine approximation to the true $\mathbf{f}$ is sufficient. We could also potentially incorporate an additional penalty term to encourage each e_r and therefore $\psi_i(\mathbf{q})$ to be small. In practice, as shown in the experiments, our method can learn a good contraction mapping with few or even no perturbations during training. Moreover, it can quickly converge to good solutions even though we always initialize from $\mathbf{q} = \mathbf{0}$ in testing, demonstrating a nice convergence property of the fixed-point model.

2.3.3 The Training and Prediction Processes

The training and prediction processes are depicted in Algorithm 1 and Algorithm 2. The training process is to learn a contextual prediction function f by favoring

Algorithm 1 The training process of the fixed-point model

```
Input: Training structures $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_N\}$ and their labelings $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\}$;
Output: The trained contextual prediction function $f$;
    1: For each $\mathbf{q}$, produce some replica $Q = \{\mathbf{q} + \boldsymbol{\delta}_r\}$ by adding random perturbations;
    2: For each node $v_i$, create the contextual feature $\overline{\mathbf{q}}_{\mathcal{N}_i}$ from the perturbed labeling $\overline{\mathbf{q}} \in Q$;
    3: Based on the feature $\mathbf{x}_i$ and $\overline{\mathbf{q}}_{\mathcal{N}_i}$, train a $f : q_i = f(\mathbf{x}_i, \overline{\mathbf{q}}_{\mathcal{N}_i}; \boldsymbol{\theta})$.
```

Algorithm 2 The testing process of the fixed-point model

```
Input: The testing structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; the trained contextual prediction function $f$; the number of iterations $\mathcal{T}$; a threshold $\varepsilon$;
Output: The labeling $\mathbf{q}$ of $\mathcal{G}$;
Initialize: $t = 1$; for each $v_i \in \mathcal{V}$, $q_i^0 = 0$;
repeat
    1: For each node $v_i$, compute the labeling $q_i^t$: $q_i^t = f(\mathbf{x}_i, \mathbf{q}_{\mathcal{N}_i}^{t-1}; \boldsymbol{\theta})$;
    2: $t = t + 1$;
until $t \geq \mathcal{T}$ or $\|\mathbf{q}^t - \mathbf{q}^{t-1}\| \leq \varepsilon$.
$\mathbf{q} = [q_1^t, q_2^t, \cdots, q_n^t]^T$.
```

fixed-point solutions (or nearly so) using some perturbations. Once learned, the contraction mapping is applied iteratively to the new structured inputs. For a novel structured input $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the labeling q_i of a node $v_i \in \mathcal{V}$ is initialized with a value. Note that q_i is not sensitive to the choice of initialization, and it is simply initialized with 0 in our experiments.

2.4 Experiments

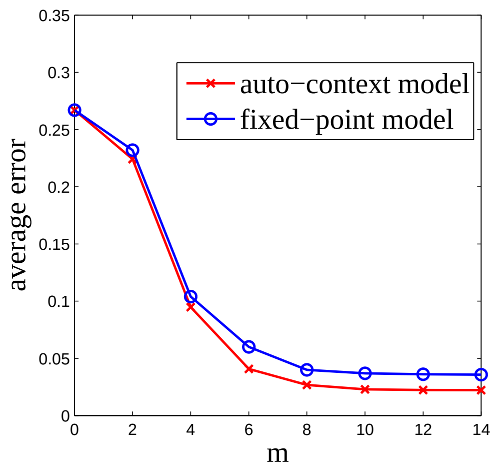
We now apply the proposed fixed-point model to the tasks of Optical Character Recognition (OCR), Part-of-Speech tagging (POS) and Hypertext (web pages) classification. The data in OCR and POS have chain structure and the average error per sequence in [NG07] is used for performance evaluation. In hypertext classification, the linking structure of the hypertext is highly non-regular and we use the average labeling error over all the testing web pages.

2.4.1 Optical Character Recognition (OCR)

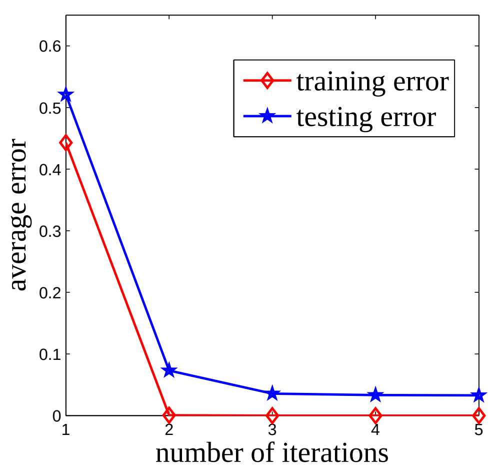
Optical character recognition involves the identification of letters in scanned texts. In this work, the benchmark dataset [TGK03] is used.

In OCR, a word corresponds to a structured input $\mathcal{V}$ and the i-th character corresponds to v_i. We use the $m/2$ characters preceding and the $m/2$ characters after v_i as its neighbors and thus m indicates the complexity of the interdependence. For each character, its pixel values are concatenated to form $\mathbf{x}_i$. In training, the lexical label y_i is encoded to a $|\mathcal{L}|$-dimensional contextual feature vector, which has value 1 only at the entry corresponding to the value of y_i. In all, a $|\mathcal{L}| \times m$-dimensional contextual feature is created. In testing, the entry corresponding to the label with the maximum score is assigned 1 at each iteration. No perturbed replicas are used in OCR and we use kernel logistic regression (KLR) [ZH01] as the contextual prediction function; at the testing process, 5 iterations are used, i.e., $\mathcal{T} = 5$.

In Fig. 2.1 (a), we compare the fixed-point model with the auto-context model. Both of the two methods use KLR with RBF kernel as the classifier. We note that the original result of the auto-context model reported in [TB10] is only 19.5% because the Harr-like features used in that work are not so effective on the OCR dataset. We compare the two models by varying m from 0 to 14. As from Fig. 2.1



(a)



(b)

Figure 2.1: (a): comparison of the testing errors on the OCR dataset by varying m; (b): the training and testing errors on the OCR dataset as $\mathcal{T}$ varies.

(a), the errors decrease monotonously as m increases. The fixed-point model performs slightly worse than the auto-context model, but it is simpler and more efficient: it takes about 7.55 minutes to train the model, while the auto-context model takes around 50 minutes because the auto-context model needs to train $\mathcal{T}$ classifiers sequentially and apply the classifiers to each of the training sequences.

In Table 2.1, we compare the fixed-point model with several state-of-the-art algorithms: SVM [CS01], SVM^{struct} [TJH05], M^{3}N [TGK03], Perceptron [COL02], KLR [ZH01], SEARN [DLM09], CRF [LMP01], HMM [RAB89], structured learning ensemble (SLE) [NG07], kernel conditional graphic model (KCGM) [CGP07] and the auto-context model [TB10]. For SVM^{struct}, M^{3}N and CRF, the results are from [NG07] with linear kernel. As from Table 2.1, with the exception of the auto-context model, the fixed-point model outperforms the state-of-the-art methods.

In [KS07], CRF and structural SVM are implemented with a different set

Table 2.1: Average errors on the OCR dataset in percentage. The results of SVM^{struct2} and CRF^2 are from [KS07]

SVM^{struct}	SVM^{struct2}	M^{3}N	SVM	Perceptron	KLR	SEARN
21.16	19.24	25.08	28.54	26.4	26.7	27.02
CRF	CRF^2	HMM	SLE	KCGM	Auto-context	Fixed-Point
32.30	19.97	23.7	20.58	5.8	2.22	3.6

of features and the performance is better than that in [NG07], see CRF^2 and SVM^{struct2} in Table 2.1. Still, the errors are much higher than that of the fixed-point model. In [TGK03], M^{3}N reports the average error per character 12.8% with cubic kernel while the average error per character of the fixed-point model is 2.13%. One may argue that if CRF, SVM^{struct} and M^{3}N were to use the contexts like those in the fixed-point model, they would generate similar results. However, it is exactly their large computational burden in taking into account long range interactions that limits their modeling ability.

In Fig. 2.1 (b), we illustrate the convergence rate of the fixed-point model, revealing that both the training and testing errors are very small after the second iteration. This suggests that we are able to train a fixed-point function that satisfies the conditions for convergence. In addition, the fixed-point model converges very quickly at the testing stage with only $2 \sim 3$ iterations.

2.4.2 Part-of-Speech Tagging (POS)

For the Part-of-Speech Tagging task, we use the POS dataset [TRE02] and comply with the training/validation/testing splits in [NG07]. For each word, $446,054$ lexical features are used. We use the L1 regularized support vector machine (SVM-L1) provided in the LIBLINEAR software package [FCH08a] as the classifier.

Table 2.2: Average errors on the POS dataset in percentage. The results of SVM^{struct2} and CRF^2 are from [KS07].

Train Size	500	1000	2000	4000	8000
SVM-L1	8.74	6.74	5.67	4.81	4.28
SVM^{struct}	8.37	6.58	5.75	4.71	4.08
SVM^{struct2}	8.38	7.15	5.63	-	-
M^{3}N	10.19	7.26	6.34	5.26	4.19
Perceptron	10.16	7.79	6.38	5.39	4.49
SEARN	10.49	8.92	7.58	6.44	5.48
CRF	16.53	12.51	9.84	7.76	6.38
CRF^2	8.84	7.08	5.83	-	-
HMM	23.46	19.95	17.96	17.58	15.87
SLE	7.71	5.93	5.14	4.19	3.67
Auto-context	8.12	6.34	5.38	4.6	3.91
Fixed-point	8.24	6.40	5.48	4.66	4.02

In our experiment, $m = 6$ is used as it gives the best results on the validation datasets. For each sequence, one perturbed replica is produced using Gaussian noise with $\boldsymbol{\delta} = 0.25$ and the contextual prediction function is trained with the perturbed replica and the original sequences.

In Table 2.2 and Table 2.3, we compare the average errors and the times to train the classifiers respectively. HMM is the most efficient in training, but its performance is poor. The fixed-point model is nearly as efficient as SVM-L1 since it needs only more feature dimensions and more training samples than SVM-L1; it is much simpler and more efficient than algorithms such as the auto-context model: on the data split of $8,000$ training sentences, it takes 2.214 hours for the auto-context model to train, while it takes only 0.192 hours for the fixed-point

Table 2.3: Training times on the POS Dataset (in hour)

Train Size	500	1000	2000	4000	8000
SVM-L1	0.0027	0.004	0.0053	0.009	0.0166
SVM^{struct}	0.11	0.21	0.38	1.7	2.2
M^{3}N	12.0	22.9	46.2	144.4	204
Perceptron	0.107	0.22	0.53	1.02	1.27
SEARN	0.035	0.043	0.053	0.096	0.13
CRF	0.53	2.33	5.4	13.3	32.7
HMM	6E-5	8E-5	1E-4	2E-4	3E-4
Auto-context	0.144	0.271	0.543	1.097	2.214
Fixed-point	0.009	0.019	0.037	0.08	0.192

model to train. The average error of the fixed-point model is slightly worse than the auto-context model but the difference is rather small. With the exception of the auto-context model and SLE, the fixed-point model outperforms the other methods. SLE performs the best but is the most complex since it is an ensemble method using about 200 different models and its training time is not listed in [NG07].

2.4.3 Hypertext Classification

Hypertext classification aims to classify the web pages based on their contents and the linking structures. We use the WebKB dataset in [CDF98] which contains web pages from 4 universities: Cornell, Texas, Washington and Wisconsin. Each page belongs to one of the 5 categories: course, faculty, student, project or other. The Bag of Words representation is used, and a codebook with $40,195$ codes is built using Rainbow [MCC96]. We compare the fixed-point model with SVM, CRF,

Table 2.4: Comparison on the WebKB Dataset.

	CRF	SVM	Auto-context	Fixed-Point
error (%)	15	22.95	16.55	16.47
train time(s)	3005	0.05	1.85	0.3

and the auto-context model. The statistics (a normalized histogram) of the labels of the in-linking and out-linking pages are used as the context feature. The fixed-point model and the auto-context algorithm both achieve small average errors when the third-order in-linking and out-linking statistics are used. For CRF, we adopt the UGM toolbox [SCH11] and use loopy belief propagation for the inference. The first order, token-independent first order, and token-independent second order feature functions are used as these feature functions give the best performance in [KS07].

The models are trained on three universities and tested on the remaining one. The average errors of the 4 universities are reported in Table 2.4. With one layer of fixed-point function, the proposed method achieves comparable result but is more efficient than the auto-context model. CRF performs the best but is much more computationally demanding.

2.5 Conclusions

In this work, we have proposed a fixed-point model for the structured labeling problem. The fixed-point model takes the labeling of the structure as both the input and the output with the assumption that the ground-truth labeling being the stable status for the function. The fixed-point model preserves the ability to capture long range contexts as in more complex layered models. A simple learning strategy is adopted and contraction conditions are analyzed. On three structured labeling problems, the fixed-point model has achieved encouraging performance and efficiency.

CHAPTER 3

Randomness and Sparsity Induced Codebook Learning

3.1 Introduction

A large number of applications in machine learning, medical image classification, and computer vision deals with the fundamental representation problem where the data are high-dimensional and live in complex manifolds. With their intrinsic and mathematical properties gradually unfolded, research in three general directions has led to significant progress on classification, recognition, and compression: (1) ensemble learning, (2) divide-and-conquer, and (3) sparse coding. More specifically, four concepts have emerged as being essential to the three directions: (1) voting, (2) randomizing, (3) partitioning, and (4) sparsity.

Ensemble learning approaches such as bagging [BRE96], boosting [FS97], and random forests [BRE01] have shown to be among the best choices for classifiers [CN06, CKY08]. The superior robustness of these ensemble methods comes from the voting/averaging of multiple independent/complementary experts (weak learners). Certain randomness in the data and feature selection stage leads to additional robustness, as shown in the random forests [BRE01] where multiple trees are learned from multiple randomly drawn subsets with the splitting criterion being locally optimal on some random features. In Extremely Randomized Trees [JEW03] and Random Projection Trees [DF08], the full data sets are used since the randomization in both feature/basis and threshold selection already provide sufficient diversities.

As real data are of high dimension and they typically do not live in a well-regularized space, assuming a Gaussian type distribution leads to limited representational power [TUR91]. When it is hard to fit a global model to the data, divided sub-problems are often easier to deal with. This is a divide-and-conquer strategy [BEN80]. In machine learning, decision tree [QUI86] is a standard approach where training data are recursively partitioned into subsets. The random forests method also has this step in training the individual tree classifier. In addition to tree node splitting, other logic rules such as And/Or can be used as well [CZL07]. The random projection tree [DF08] also has recursive data partition based on randomly generated bases.

More recently, sparse representations such as compressed sensing [CT05] and LASSO [TIB96] have gained a great deal of popularity. One message emerging from sparse representation is that high-dimensional data within intrinsic lower dimension can be well represented by sparse samples of high dimension. The robustness of the sparse representation often assumes a subspace of certain regularity, e.g. well-aligned data [WYG09].

In this work, we tackle the problem of codebook learning for high dimensional visual data. Inspired by the above observations, we propose a randomized forest sparse coding (RFSC) method. Given a large set of visual data, we train an ensemble of random splitting/projection trees (when we are not sure about the form of the whole data population, it is desirable to perform random partition with certain local optimality); for each leaf node in the tree, we learn a set of bases to best represent the data with sparse coefficients. The overall codebook is a collection of all the bases from all the tree leaves. RFSC carries the ideas of voting, randomizing, partitioning, and sparse coding in a natural way. It's applicable to applications such as natural image classification, and modern cancer diagnosis.

Modern cancer diagnosis largely benefits from high resolution histopathology images, which provide distinctive and reliable cues for discriminating abnormal tissues from normal ones. Therefore, automatic recognition and analysis of cancer in histopathology images are very important assistant means for doctors. In this work, we verify our algorithm on a collection of colon cancer images. As shown in the experiment section, promising results are obtained.

3.2 Related Work

As we have discussed, our approach is inspired by the literature in ensemble learning [BRE96, FS97, BRE01], divide-and conquer approaches [BEN80, QUI86, CZL07], and sparse representation [CT05, TIB96, WYG09, MBPar]. Two types of work are particularly related to our approach: tree based splitting/projection methods, e.g., Extremely Randomized Trees [JEW03] and Random Projection Trees [DF08], and sparse coding based codebook learning techniques [YYG09a, LSP06a, GTC10].

Extremely Randomized Tree (ERT) [JEW03] is a variant of random forest. ERTs randomize both the feature selection and the quantization threshold searching process, making the trees less correlated. When used for visual codebook learning (ERC-Forest) in [MNJ08a], the generated trees are not treated as an ensemble of decision trees, instead, they are referred to as an ensemble of hierarchical spatial partitioners. The samples (image patches) in each leaf node are assumed to form a small cluster in the feature space. The leaves in the forest are uniquely indexed and serve as the codes for the codebook. When a query sample reaches a leaf node, the index of that leaf is assigned to the query sample. A histogram is formed by accumulating the indices of the leaf nodes, which serves as a Bag of Words (BOA) representation. Similar to ERC-Forest, [SJC08] introduces a semantic texton forest using ERT to perform image classification and segmentation.

Random Projection Tree [DF08] is a variant of k-d tree. The k-d tree splits the data set along one coordinate at the median and recursively builds the tree. Though widely used for spatial partitioning, it suffers from the curse of dimensionality problem. Based on the realization that, high dimension data often lies on low-dimensional manifold, RPT splits the samples into two roughly balanced sets according to a randomly generated direction. This randomly generated direction approximates the principal component direction, and can adapt to the low dimensional manifold. The RPT naturally leads to tree-based vector quantization and an ensemble of RPTrees can be used as a codebook.

We use Extremely Randomized Trees/Random Projection Trees to partition the samples. But instead of splitting the samples till we cannot split any more, we stop early according to certain criterion and find some bases that can best reconstruct all the samples in that node. These bases serve as codes of the codebook.

There are already some methods using sparse coding for codebook learning. In [YYG09a], the authors generalize vector quantization to sparse coding, and construct the histogram using multi-scale spatial max pooling. Each patch can be assigned to several (sparse) codes, and thus the reconstruction error can be reduced. Also, this method extends the Spatial Pyramid Matching method [LSP06a] to a linear SPM kernel. In [GTC10], Laplace sparse coding preserves the consistency in the sparse representation and alleviates the problem in [YYG09a] that similar patches may be assigned to different codes. In [WYY10a], a locality-constrained linear coding scheme is proposed that utilizes the locality constraints to project descriptors to their local-coordinate system. This scheme can preserve the property of local smooth sparsity. Compared with these methods, the advantages of RFSC is obvious. One advantage is the efficiency. Utilizing techniques such as ERT and RPT, the sparse coding is performed only in subspaces and the computational burden is greatly reduced.

The second advantage is the potential promotion of the discriminative ability. The label information can easily be used into the tree splitting process (ERT) and the codebook created could have more discriminative power.

3.3 Randomized Forest Sparse Coding

3.3.1 Problem Formulation

Suppose we are given a set of training data $S = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{x}_i \in \mathbb{R}^D$ (in a supervised setting, each $\mathbf{x}_i$ is also associate with a label $y_i \in \mathcal{Y} = \{0, ..., K\}$ and thus $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$), our goal is to learn a codebook (set of basis) $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^m$ and $\mathbf{b}_i \in \mathbb{R}^D$ such that

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{w}} \sum_{i=1}^n \left\| \mathbf{x}_i - \sum_{j=1}^m w_{ij}\mathbf{b}_j \right\|_2^2 \\ &s.t.\ \ \forall i, \sum_j |w_{ij}| \leq \tau \end{aligned} \tag{3.1}$$

The first term in Eqn. (3.1) minimizes the reconstruction error and the second term gives the sparsity constraints on the reconstruction coefficients. Eqn. (3.1) actually includes two coupled optimization problems: (1) given $\mathbf{w}$, find the optimal codebook $\mathbf{B}$; (2) given a codebook $\mathbf{B}$, find the best reconstruction coefficients $\mathbf{w}$. A similar formulation appears in [YYG09a].

After an optimal basis set $\mathbf{B}^*$ is found, for a new sample $\mathbf{x}$, we can compute its reconstruction coefficients $\mathbf{w}$ via:

$$\begin{aligned} &\min_{\mathbf{w}} \left\| \mathbf{x} - \sum_{j=1}^m w_j\mathbf{b}_j \right\|_2^2 \\ &s.t.\ \ \sum_j |w_j| \leq \tau \end{aligned} \tag{3.2}$$

The vector $\mathbf{w}$ can be used to characterize the sample $\mathbf{x}$. In codebook learning, each $\mathbf{b}_j$ serves as a code, and the reconstruction coefficients with respect to the codes are pooled to form a histogram.

In Eqn. (3.1), the norm of $\mathbf{b}_j$ can be arbitrarily large, making w_{ij} arbitrarily small. Further constraints should be made on $\mathbf{b}_j$. In our work, we make a reasonable constraint that all the basis in the codebook should be from the training set S, i.e., $\mathbf{B} \subset S$. With this constraint, Eqn. (3.1) can be transformed into

$$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j=1}^{n} w_{ij} v_j \mathbf{x}_j \right\|_2^2 \tag{3.3}$$

$$s.t. \;\; \sum_j v_j \leq m, v_j \in \{0, 1\}$$

$$\forall i, \;\; \sum_j |w_{ij}| \leq \tau \tag{3.4}$$

Here, v_j serves as an indicator value $\in \{0, 1\}$ and $\mathbf{B} = \{\mathbf{x}_j : \mathbf{x}_j \in S, \; v_j = 1\}$. Eqn. (3.3) is seemingly more complex than Eqn. (3.1) with the introduction of $\mathbf{v}$. However, it can be solved more efficiently since the search space for the basis is greatly reduced.

Learning a codebook of size greater than e.g. $5,000$ from tens of thousands of samples is computationally demanding. However, recent research reveals that data of real-world complexity often live in complex manifolds. As motivated before, we could perform a divide-and-conquer strategy to partition the data space into local subspaces. Within a subspace, it is then much more efficient to learn bases for a sparse representation.

3.3.2 Randomized Forest Data Partition

In this section, we take the Extremely Randomized Tree (ERT) [JEW03] and Random Projection Trees (RPT) as examples to illustrate the data projection process. Both ERT and RPT partition the samples recursively in a top-down manner. ERT adopts the label information and uses normalized Shannon entropy as the criterion to select features. RPT is unsupervised and it does not need any label information; it splits the data via a hyperplane normalized to each individual randomly generated projection bases.

3.3.2.1 Discriminative Partition via Extremely Randomized Tree

Given a labeled sample set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, ERT proceeds by randomly selecting a subset of features from the feature pool $\{f_i, 1 \leq i \leq D\}$. For each selected feature f_i, a threshold θ_i is sampled according to a uniform distribution (in [MNJ08a], a Gaussian distribution adapted to the feature values in that dimension). Based on the features selected and thresholds sampled, boolean tests $\{T_i : \mathbf{x}(i) < \theta_i\}$ can be used to split the set S. If $T_i = \text{true}$, $\mathbf{x}$ goes to the left branch S_1, else, $\mathbf{x}$ goes to the right branch S_2.

To select the best boolean test for splitting, the normalized Shannon entropy was used:

$$Score(S, T_i) = \frac{2 \cdot I_{\mathcal{Y},T_i}(S)}{H_{\mathcal{Y}}(S) + H_{T_i}(S)} \tag{3.5}$$

where, $I_{\mathcal{Y},T_i}(S) = H_{\mathcal{Y}}(S) - \sum_{p=1}^{2} \frac{n_p}{n} H_{\mathcal{Y}}(S_p)$. $I_{\mathcal{Y},T_i}(S)$ is the information gain, a non-negative scalar denoting the uncertainty reduced by the test T_i. $H_{\mathcal{Y}}(S) = -\sum_{y \in \mathcal{Y}} \frac{n_y}{n} \log_2(\frac{n_y}{n})$ denoting the entropy of class distribution of the original set S. $H_{T_i}(S) = -\sum_{p=1}^{2} \frac{n_p}{n} \log_2(\frac{n_p}{n})$ denotes the entropy for the test T_i that splits the data into two branches. The T_i with the largest $Score(S, T_i)$ is selected.

The use of $H_{T_i}(S)$ as a normalization term in Eqn. (3.5) was first introduced in [QUI86] to resolve the bias problem: the criterion $I_{\mathcal{Y},T}(S)$ will be biased towards the attributes leading to more branches. In codebook learning, since we are using binary splitting, this bias problem is not a concern. In fact, the use of $H_{T_i}(S)$ as a normalization term will favor uneven splitting, making the forest more unbalanced.

In our randomized forest sparse coding scheme (RFSC), it is desirable to have balanced trees, so we use a slightly modified form of Eqn. (3.5):

$$Score(S, T_i) = \frac{2 \cdot I_{\mathcal{Y},T_i}(S)}{H_{\mathcal{Y}}(S) + 1 - H_{T_i}(S)} \tag{3.6}$$

Since $H_{T_i}(S)$ is a concave function and it achieves the maximum value 1 when the numbers of samples in S_1 and S_2 are the same, this criterion can make the trees more balanced.

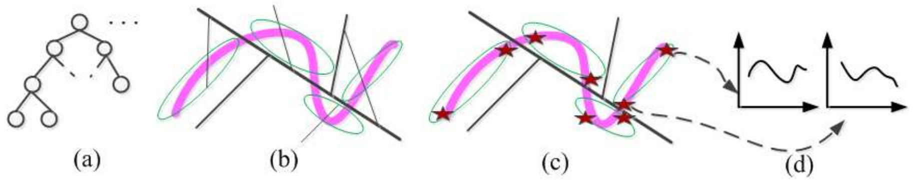


Figure 3.1: Illustration of the idea of RFSC using Random Projection Tree (best viewed in color). (a) The forest consists of ensemble of random projection trees; (b) The spatial partition of the dataset by one tree (A copy from [FDK07]). A cell stands for a leaf node. The width of the separation line indicates the level of the tree. (c) For RFSC, it does not build the tree to fine level. At certain level when local manifold structures are found, bases (indicated by the red stars) are learned for the local structure in each cell. (d) For the samples in each cell, their reconstruction coefficients with respect to the bases are different.

3.3.2.2 Unsupervised Splitting via Random Projection Tree (RPT)

At each node, RPT chooses a random unit projection $\mathbf{b} \in \mathbb{R}^D$, and splits the samples into two roughly equal-sized sets. The random projection and thresholding also serve as a type of boolean test. We use the splitting criterion as

$$T := \mathbf{x}^T\mathbf{b} \leq \left(\text{median}(\mathbf{z}^T\mathbf{b} : \mathbf{z} \in S) + \delta\right).$$

Here δ is a random perturbation that adapts to the structure of S. Splitting around the median value makes the splitting balanced while the perturbation δ introduces certain randomness [DF08].

Since RPTs can automatically adapt to the low dimensional manifold of the dataset S, the samples in the leaf nodes observe local subspaces. The local structures of all the leaf nodes thus collectively comprise the global structure of the data set S (Fig. 3.1 (b)).

3.3.2.3 Basis Pursuit at the Leaf Nodes

Both ERT and RPT build the trees to the fine scale and use the leaf nodes as the codes. Instead of building the trees of very deep level, RFSC stops at some relatively higher level (e.g., when the number of samples is less than M). At such nodes, the local manifold structure is assumed to be relatively simple and regularized. RFSC seeks a set of bases to sparsely represent the subspaces at those nodes. This process can be illustrated using Random Projection Tree in Fig. 3.1 in which a visualization is displayed and RPT tends to split the data along the principal component direction (Fig. 3.1 (b)). For RFSC, when the local structure is relatively regularized, it seeks some bases (the red stars) to sparsely represent the local subspace. Different from RPT or ERT that use the mean of the local subspace or a single index to represent the cell, the information conveyed via the reconstruction coefficients with respect to each basis (Fig. 3.1 (d)) is richer and more informative. Note that the bases in different clusters could be spatially close to each other. As an illustration, see the two bases on the bottom right in Fig. 3.1(c). From this point of view, the number of basis and the redundancy would increase. However, multiple graphs [UST09] could help to smooth the boundaries of the overall data representation, and thus, lead to enhanced overall performance. Also, according to Theorem 1 in the justification part, the total number of bases in all the leaf nodes is bounded. Since at each node when the splitting process stops, there are generally $80 \sim 200$ samples (depending on the codebook size) and $3 \sim 10$ bases, the computational overhead of subspace learning is not significant compared with directly pursuing basis from the entire sample set.

3.3.3 Optimization Scheme

The constraint that $v_j \in \{0, 1\}$ makes Eqn. (3.3) a hard problem. In this subsection, we present two schemes to solve this optimization problem. The first one is to relax v_j to a real value and use coordinate descent algorithm to optimize on $\mathbf{w}$ and $\mathbf{v}$ iteratively. The second one is a greedy pursuit approach that selects the bases one by one.

Convex Relaxation The first optimization scheme is to relax the values of v_j to real numbers and use ℓ^1 constraint $\sum_j |v_j| \leq m$ instead of ℓ^0 like constraint in Eqn. (3.3). Putting this constraint as a regularization term, we can transform this problem into an equivalent form:

$$\frac{1}{2}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \sum_{j=1}^{n} w_{ij}v_j\mathbf{x}_j\right\|_2^2 + \lambda_1 \sum_{i,j}|w_{ij}| + \lambda_2 \sum_j |v_j| \tag{3.7}$$

Here, $v_j \in \mathbb{R}$. λ_1 and λ_2 are regularization parameters that make the trade-offs between the residue and the norms of the weight vectors.

There are two sets of variables $\mathbf{w}$ and $\mathbf{v}$ in Eqn. (3.7). To optimize Eqn. (3.7), we adopt an EM-like algorithm that iterates by fixing one set of variables and optimize on the other set using coordinate descent algorithm [FHH07].

By fixing $\mathbf{w}$, we can get the updated value $\tilde{v}_j$ of v_j as:

$$\tilde{v}_j = \frac{\text{shrink}\left(\sum_i w_{ij}\beta_{ij}, \lambda_2\right)}{\sum_i w_{ij}^2 \mathbf{x}_j^T \mathbf{x}_j} \tag{3.8}$$

Here, $\beta_{ij} = \mathbf{x}_j^T \left(\mathbf{x}_i - \sum_{k\neq j} w_{ik}v_k\mathbf{x}_k\right)$. shrink$(f, \lambda)$ is an operator to shrink the value of f toward 0:

$$\text{shrink}(f,\lambda) = \begin{cases} f-\lambda & \text{if} \quad f > \lambda \\ \quad 0 & \text{if} -\lambda \leq \lambda \leq \lambda \\ f+\lambda & \text{if} \quad f < -\lambda \end{cases} \tag{3.9}$$

By fixing $\mathbf{v}$, we get the updated value $\tilde{w}_{ij}$ of w_{ij} as:

$$\tilde{w}_{ij} = \frac{\text{shrink}\left(v_j\beta_{ij}, \lambda_1\right)}{v_j^2 \mathbf{x}_j^T \mathbf{x}_j} \tag{3.10}$$

For the sake of efficiency, in practice, instead of using the pathwise coordinate descent algorithm [FHH07] that sweeps all the variables sequentially,we adopt an adaptive and greedy sweeping version [LO09] that sweeps the variable that decreases the objective function most at each iteration. For w_{ij} and its updated value $\tilde{w}_{ij}$, the decrease of the objective function is

$$\begin{aligned} \Delta R_{w_{ij}} = & \ \tfrac{1}{2} v_j^2 \left\| \mathbf{x}_j \right\|_2^2 (w_{ij} - \tilde{w}_{ij}) \left(w_{ij} + \tilde{w}_{ij} - \tfrac{2 v_j \beta_{ij}}{v_j^2 \| \mathbf{x}_j \|_2^2} \right) \\ & + \lambda_1 (|w_{ij}| - |\tilde{w}_{ij}|) \end{aligned} \tag{3.11}$$

For v_j and its updated value $\tilde{v}_j$, the decrease of the objective function is

$$\begin{aligned} \Delta R_{v_j} = & \ \textstyle\sum_{i=1}^{n} \tfrac{1}{2} w_{ij}^2 \left\| \mathbf{x}_j \right\|_2^2 (v_j - \tilde{v}_j) \left(v_j + \tilde{v}_j - \tfrac{2 w_{ij} \beta_{ij}}{w_{ij}^2 \| \mathbf{x}_j \|_2^2} \right) \\ & + n\lambda_2 (|v_j| - |\tilde{v}_j|) \end{aligned} \tag{3.12}$$

At each iteration, the updating rule is to select the variable that leads to the largest decrease in the objective function. Thus, when fixing $\mathbf{v}$ to optimize on $\mathbf{w}$,

$$w_{ij}^* = \arg\max_{w_{ij}} \Delta R(w_{ij}) \tag{3.13}$$

and when fixing $\mathbf{w}$ to optimize on $\mathbf{v}$

$$v_j^* = \arg\max_{v_j} \Delta R(v_j) \tag{3.14}$$

In our experiment, the adaptive and greedy sweeping proves efficient for practical use. After the optimization process converges, we rank the samples according to their v_j. The first m samples with the largest non-zero v_j are selected as the basis.
Greedy Pursuit Approach Starting from an empty basis collection, the greedy pursuit approach selects the basis one by one. Suppose some l samples $\mathbf{B}_l = \{\mathbf{x}_{s_i}, 0 \le i \le l, 1 \le s_i \le n\}$ have been selected from the n samples, i.e., $v_{s_i} = 1$. To select the $(l+1)$th basis, we optimize the following function:

$$\begin{aligned} s_{l+1} = & \quad \min_{k \notin \{s_i\}} \frac{1}{2} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in \{s_i\}} w_{ij} \mathbf{x}_j - w_{ik} \mathbf{x}_k \right\|_2^2 \\ & + \lambda_1 \sum_{i=1}^{n} \sum_{j \in \{s_i\}} |w_{ij}| + \lambda_1 \sum_{i=1}^{n} |w_{ik}| \end{aligned} \tag{3.15}$$

According to Eqn. (3.15), the sample that reconstructs all the n patches together with the first l selected basis is selected as the $(l+1)$th basis.

The greedy approach finds suboptimal solution to Eqn. (3.3). But it's more efficient than the convex relaxation approach, and in practice, we find that its performance is comparable with the convex relaxed solution. Thus in some of our experiments, we only use this greedy approach.

3.3.4 Theoretical justification

In this section, we give some theoretical justification to our approach. Our institution is to show that the three steps in randomized forest sparse coding: (1) ensemble of trees, (2) randomized projection tree, and (3) sparse coding leads to the same complexity level in the number of basis as to the original data.

Given $S = \{\mathbf{x}_i, i = 1..n\}$ with $\mathbf{x}_i \in \mathbb{R}^D$, assume that $\mathbf{x}_i$ lives in the intrinsic lower dimension $d \ll D$. It can be seen that the number of basis needed to reconstruct S is bounded. Following the definition of Assouad dimension [ASS83] [DF08]:

Definition: For any point $\mathbf{x} \in \mathbb{R}^D$ and $r > 0$, let $B(\mathbf{x}, r) = \{||\mathbf{x}-\mathbf{z}|| \leq r\}$ denote the closed ball of radius r centered at $\mathbf{x}$. The Assouad dimension of $S \in \mathbb{R}^D$ is the smallest integer d such that for any ball $B(\mathbf{x}, r) \in \mathbb{R}^D$, the set $B(\mathbf{x}, r) \cap S$ can be covered by 2^d balls of radius $r/2$.

Theorem 1 *The number of basis needed to reconstruct S by Randomized Forest Sparse Coding (RFSC) is $O(2^{d \log d})$.*

Proof:

Fixing radius r, suppose we want to create a codebook such that each basis function covers $r/2$, a size of $O(2^d)$ codebook is required to cover the entire dataset S, according to the definition of Assouad dimension.

The main result in [DF08] shows that $O(d \log d)$ levels of a random projection/partition tree would reach cells with radius $r/2$. Therefore, the number of cells is $O(2^{d \log d})$. Suppose there are k trees in the forest, and in each leaf node, l basis are found, then the number of the basis becomes $O(kl2^{d \log d})$. As k and l are generally small and can be kept constant, the bound still reduces to:

$$O(2^{d \log d}).$$

Although RFSC slightly increases the size of the codebook compared to $O(2^d)$, since d is generally small ($d \ll D$), this is reasonably bounded.

3.4 Experiments

To evaluate the effectiveness of the proposed codebook learning algorithm, we conducted extensive classification experiments on a collection of cancer images and a variety of natural image datasets: Graz-02 image set, the INRIA Horse dataset, and the PASCAL 2005 image set.

As the baselines, we obtained the source code for ERC-Forest from the authors of [MNJ08a] and implemented the RPTs according to [DF08]. In our experiments, the feature vectors are used without any normalization, which is sometimes done in subspace learning and sparse coding (we found that performing normalization does not affect the overall performance in the experiments reported here). For each leaf node, 5 bases are learned. For the Graz-02 image set, $\lambda_1 = 2$ and $\lambda_2 = 6$, while for the INRIA Horse dataset and the PASCAL 2005 image set, $\lambda_1 = 15$ and $\lambda_2 = 6$. To solve the subspace learning problem via sparse coding defined in Eqn. (3.3), 10 iterations between $\mathbf{w}$ and $\mathbf{v}$ are enough to find a good sparse solution.

In the following, we use RFSC to denote subspace learning via sparse coding under Extremely Randomized Trees; RPT-SC denotes subspace learning on

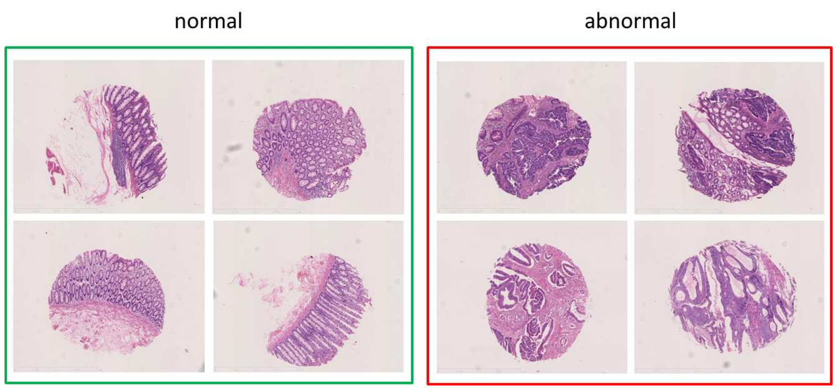


Figure 3.2: Cancer image examples. The images in the green box are normal samples. i.e. there are no cancerous cells. The images in the red box are abnormal samples, i.e. there are cancerous cells.

Random Projection Trees. For RFSC and RPT-SC, the postfix "-Cvx" refers to using the convex relaxation version and "-Gdy" regards to using the greedy basis pursuit version. For the classification task of Cancer Images, the performance is measured using the Area under the curve of the ROC curve, while for natural image classification, the performance is measured using the classification accuracy at the Equal Error Rate and the reported accuracies are the averages of 10 rounds of execution. As can be seen from the experiments, our method achieves comparable or superior performance with the alternatives.

3.4.1 Experiments on Cancer Images

Dataset: We used a histopathology image data set with 60 colon images (30 cancer images and 30 non-cancer images). Some example images from this dataset are shown in Fig. 3.2. The images are in the resolution of 1280×1024. All the images are labeled as cancer or non-cancer by two pathologists independently. If disagreement happens for a certain image, these two pathologists together with a third senior pathologist will carefully examine and discuss until final agreement.
Experimental Setup: Before feature extraction, the original images are down-

sampled with a factor of 2. Since no obvious spatial regularities are observed from the images (Fig. 3.2), we didn't densely compute local features and construct Bag-of-Features (BOF) vectors. Instead, we randomly sample $N = 200$ local patches (32×32) for each image. Each patch is represented by Lab color histogram, Local Binary Pattern [OPM02], and SIFT [LOW04]. For the proposed method, each patch is encoded by the proposed coding schemes RFSC or RPT-SC; for the baseline, we use the raw feature. Random Forests [BRE01] is adopted as the strong classifier for its simplicity and high performance. The overall classification score of an image is the mean of the scores of all the patches. Half of the images in the dataset are chosen randomly for training and the rest for testing. We run the experiments 5 times for each method and report the averaged performance. For RFSC and RPT-SC, the convex relaxation versions are used. We compare the Area under the ROC curve between RPT-SC, RFSC, ERC-Forest, RPT, and raw feature. The Area Under Curve (AUC) for the methods are RPT-SC 0.98, RFSC 0.987, RPT 0.927, ERC-Forest 0.95, and raw feature 0.967 respectively; our method performs better than the alternatives.

3.4.2 Experiments on Natural Images

The reconstruction coefficients are pooled in the natural image classification task. To pool the reconstruction coefficients, unless otherwise stated, max-pooling is used as in [YYG09a]. The pooled reconstruction coefficients of the trees are concatenated to form a histogram leaving the voting process till the classification step. SVM is used as the classification model, and the linear kernel is used. To understand better the behavior of the competing codebook learning algorithms, in all the following image classification experiments, we do not include the adaptive saliency map process. This makes the image classification performance of ERC-Forest slightly worse than that reported in [MNJ08a]. However, this performance degeneration is understandable and in accordance with the case illustration in

[MNJ08a]. Focusing on the core codebook learning part helps to better validate the underlying benefits of our method against the competing algorithms.

GRAZ-02 dataset [OPF06] GRAZ-02 image set consists of three object categories and one counter-category: Bicycles (365 images), Cars (420 images), Person (311 images) and None (380 images). For each category, the categorization task is to distinguish the object category from the counter-category, None. Similar to [MNJ08a], we also pick the two hardest cases: Cars vs. None and Bikes vs. None. Patches are sampled from the images and 768-D wavelet feature vectors are used as the descriptors.

To make a direct comparison with [MNJ08a] and [OPF06], we conduct the experiment according to the setting in [MNJ08a]: the first 300 images of each category are used and 5 trees are trained. We use the greedy version of RFSC and vary the codebook size from 5000 to 9000. From Table 3.1 and Table 3.2, we observe that, RFSC-Gdy performs better than ERC-Forest and the method in [OPF06]. Although RPT-SC-Gdy does not outperform ERC-Forest, it still performs better than RPT on both of the two cases. RFSC-Gdy outperforms RPT-SC, indicating the promotion of the discriminative ability by introducing the label information in the divide-and-conquer process. Even without the adaptive saliency map process, the accuracy (83.9%) of RFSC-Gdy on the case of Bikes vs. None approaches that reported in [MNJ08a] (84.4%). Note that we only use 5 bases in each leaf node to represent a 768 dimensional feature space; the large improvement in classification accuracy not only proves the relative regularity in the local subspaces, but also supports the formulation in Eqn. (3.1) and the effectiveness of the sparse representation.

We also conduct the experiments using all the images instead of the first 300 images. Average-pooling is adopted here and the results are reported in Table 3.3 and Table 3.4. The performance of the two optimization schemes is similar: for the case of Cars vs. None, RFSC-Gdy achieves the best accuracy 75.5% and for

Table 3.1: Comparison of the accuracy on the case of Cars vs. None in the GRAZ-02 images [OPF06].

size of codebook	5000	6000	7000	8000	9000
[OPF06]			70.5%		
ERC-Forest	71.3%	73.5%	74.5%	74.7%	74.8%
RPT	66.5%	66.6%	65.3%	67.7%	66.9%
RFSC-Gdy	73.4%	74.3%	**75.7%**	74.9%	74.3%
RPT-SC-Gdy	68%	69.8%	69%	69.5%	68.2%

Table 3.2: Comparison of the accuracy for Bikes vs. None in the GRAZ-02 images [OPF06].

size of codebook	5000	6000	7000	8000	9000
[OPF06]			77.8%		
ERC-Forest	78.8%	78%	78.5%	78.5%	78.5%
RPT	73.3%	74.3%	74.1%	75.1%	74.4%
RFSC-Gdy	80.7%	**83.9%**	80.8%	81.3%	80%
RPT-SC-Gdy	76.5%	76.8%	76.1%	76.7%	76%

Table 3.3: Comparison of the accuracy using all the images for Cars vs. None in the GRAZ-02 images [OPF06].

size of codebook	5000	6000	7000	8000	9000
ERC-Forest	67.2%	67%	68.6%	68.8%	71.3%
Leaf-Kmeans	68.2%	70.9%	73%	72.6%	73.2%
RFSC-Cvx-1tree	72.6%	72.2%	71.4%	75%	75%
RFSC-Cvx	75%	75%	73.7%	73.1%	75.2%
RFSC-Gdy	74.3%	75.5%	74.5%	74.8%	**75.5%**

Table 3.4: Comparison of the accuracy using all the images for Bikes vs. None in the GRAZ-02 images [OPF06].

size of codebook	5000	6000	7000	8000	9000
ERC-Forest	77.8%	78.3%	78.3%	79.1%	78.8%
Leaf-Kmeans	75.1%	74.4%	79.7%	78.7%	79.5%
RFSC-Cvx-1tree	77.8%	78.2%	78.6%	79.5%	79.5%
RFSC-Cvx	80%	82.2%	**82.6%**	81.4%	81.8%
RFSC-Gdy	81.5%	80.3%	81.5%	80.8%	80.9%

Bikes vs. None, RFSC-Cvx achieves the best accuracy 82.6%. RFSC-Cvx-1tree refers to using one randomized tree instead of the forest, an ensemble of trees. It performs worse than RFSC. This justifies the benefit of using ensembles and is in accordance with the spirit in ensemble learning: the randomized partition process provides sufficient diversities among codes of the forest, and the concatenated codebook produces better and more robust results via voting.

We do not compare RFSC and RPT-SC with directly performing dictionary learning on the image classification task since solving Eqn. (3.1) directly when

$m = 5,000$ or $9,000$ is time consuming. However, benefiting from the divide-and-conquer process, it takes less than 1 hour for RFSC and RPT-SC to build a forest with 5 trees and $9,000$ codes. This improvement in efficiency stems from seeking a small amount of bases from hundreds of patches instead of seeking thousands of bases from tens of thousands of training patches. Other efficient algorithms such as [LBR07] can be used to solve Eqn. (3.1), but the conclusion of the improvement in efficiency induced by the divide-and-conquer process still holds. RFSC and RPT-SC are also very efficient at the testing stage. It takes about 0.5 second to process an image and pooling the reconstruction coefficients. As a comparison, it would take around 30 seconds for K-Means to assign patches to the codes for an image when the feature vector is of dimension 768 and the codebook size K is $5,000$.

INRIA Horse Dataset [FJS07a] INRIA horse dataset contains 170 horses taken from the Internet with different sizes and poses. The training/splitting ratio of this dataset and the size of the codebook were not reported in [MNJ08a], so we randomly selected 85 horse images for training and varied the size of codebook from $5,000$ to $9,000$. The SIFT descriptor is used to describe the patches and we used the dense SIFT implementation in [VF08]. The greedy pursuit approach was used. In Table 3.5, we report the best accuracy of each method and the size of the codebook at which the best accuracy is achieved. From Table 3.5 we observe that, RFSC-Gdy performs better than ERC-Forest and RPT-SC-Gdy performs better than RPT. The performance of ERC-Forest has the potential to be improved if the size of the codebook increases. We did not carry out the experiment, since even without estimating the saliency map and with small codebook, RFSC-Gdy has already achieved better result (85.9%) than that reported in [MNJ08a] (85.3%).

PASCAL 2005 image set [EZ05] We also compare our method with ERC-Forest on PASCAL 2005 image set. The results are shown in Table 4.1. RFSC-Gdy achieves better results on all of the 4 categories than ERC-Forest.

Table 3.5: Comparison of the accuracy on the INRIA Horse dataset [FJS07a].

method	ERC-Forest	RPT	RFSC-Gdy	RPT-SC-Gdy
Accuracy	79.2%	75.7%	**85.9%**	80.4%
size of Codebook	9000	7000	5000	8000

Table 3.6: Comparison of the accuracy on PASCAL 2005 image set [EZ05].

method	motobikes	cars	bikes	person
ERC-Forest	96%	95%	89%	90.9%
RFSC-Gdy	**96.4%**	**95.3%**	**90.6%**	**91.4%**

3.5 Conclusion

In this work, we have introduced a codebook learning method called randomized forest sparse coding that integrates three concepts: ensemble, divide-and-conquer and sparse coding. Justifications for the effectiveness and efficiency of our method are also provided. The proposed scheme is applied to both the Cancer Image Classification and natural image classification and observes significant improvement in performance.

CHAPTER 4

Harvesting Mid-level Visual Concepts from Large-scale Internet Images

4.1 Introduction

The inventions of robust and informative low-level features such as SIFT [LOW04], HOG [DT05], and LBP [OPH96] have been considered as one of the main advances/causes for the recent success in computer vision. Yet, one of the most fundamental issues in vision remains to be the problem of "representation", which affects an array of applications, such as image segmentation, matching, reconstruction, retrieval, and object recognition.

Beyond low-level features, obtaining effective mid-level representations has become increasingly important. For example, there have been many recent efforts made along the line of attribute learning [FEH09, PG11a, LSX10]. These approaches, however, are mostly focused on supervised or active learning where a considerable amount of human efforts are required to provide detailed manual annotations.The limitations to the previous supervised attribute learning methods are thus three-fold: (1) accurate data labeling is labor-intensive to obtain, (2) the definition of attributes is often intrinsically ambiguous, (3) the number of attributes and training images are hard to scale. Some other methods in which detailed manual annotations are not required (e.g. classemes [HDF12]) however are not designed to build a dictionary of mid-level representations.

In this work, we propose a scheme to build a path from words to visual concepts; using this scheme, effective mid-level representations are automatically exploited from a large amount of web images. The scheme is inspired by the following observations: (1) search engines like Google and Bing have a massive number of well organized images; (2) using text-based queries, such as “bike”, “bird”, “tree”, allows us to crawl images of high relevance, good quality, and large diversity (at least for the top-ranked ones); (3) the multiple instance learning formulation [ATH02] enables us to exploit common patterns from retrieved images, which have a high degree of relevance to the query words; (4) saliency detection [FWT11] helps to reduce the search space by finding potential candidates. The main contributions of this work thus include the following aspects: (1) we emphasize the importance of automatic visual concept learning from Internet images by turning an unsupervised learning problem into a weakly supervised learning approach; (2) a system is designed to utilize saliency detection to create bags of image patches, from which mixture concepts are learned; (3) consistent and encouraging results are observed by applying the learned concepts on various benchmark datasets.

Visual attribute learning has recently attracted a lot of attention. However, many existing algorithms were designed as supervised approaches [FEH09, PG11a, LSX10, PG11b, LSX10], preventing them from scaling up to deal with a large number of images.

A term, “classeme”, was introduced in [TSF10] which also explores Internet images using word-based queries; however, only one classeme is learned for each category and the objective of the classeme work is to learn image-level representations. Instead, our goal here is to learn a dictionary of mid-level visual concepts for the purpose of performing general image understanding, which goes out of the scope of classeme [TSF10] as it is computationally prohibitive for [TSF10] to train on a large scale.

A recent approach [SGE12] learns “discriminative patches” in an unsupervised manner. However, [SGE12] learns discriminative patches while we focus on dictionary learning for the mid-level representations; [SGE12] uses an iterative procedure, while our method adopts saliency detection, miSVM and K-means in a novel way; in addition, our method significantly outperforms [SGE12] with a relative 37% improvement on the MIT-Indoor scene dataset, on which both the approaches have been tested. In [LRM12], high-level features are built from large scale Internet images with nine layers of locally connected sparse autoencoder; however, their auto-encoder approach is much more complex than the scheme proposed in this work. In [ZWW12], saliency detection is utilized to create bags of image patches, but only one object is assumed in each image for the task of object discovery. Although multiple clusters are learned in [XZC12], its goal is to identify a few cancer patterns for medical image segmentation; in addition, the lack of explicit competition among clusters leads to poor results in our problem. In [HSH13], compositional sparse codes are learned for natural image representation, and in [SZ12], hybrid image templates consisting of sketches, texture / gradients, flat areas, and colors are learned. Both of the two representations are generative and visually meaningfully, but they take a long time to learn. In terms of large-scale natural images, ImageNet [DDS09] is shown to be a great resource. Here, we find it convenient to directly crawl images from the search engines using word-based queries.

4.2 Automatic Visual Concept Learning

Starting from a pool of words, we crawl a large number of Internet images using the literal words as queries;patches are then sampled and visual concepts are learned in a weakly supervised manner. The flow chart of our scheme is illustrated in Fig. 4.1. Following our path of harvesting visual concepts from words, many algorithms can be used to learn the visual concepts. In this work, we adopt a simple scheme, using the max-margin formulation for multiple instance learning in [ATH02] to automatically find positive mid-level patches; we then create visual concepts by performing K-means on the positive patches. The visual concepts learned in this way are the mid-level representations of enormous Internet images with decent diversity, and can be used to encode novel images and to categorize novel categories. In the following sections, we introduce the details of our scheme.

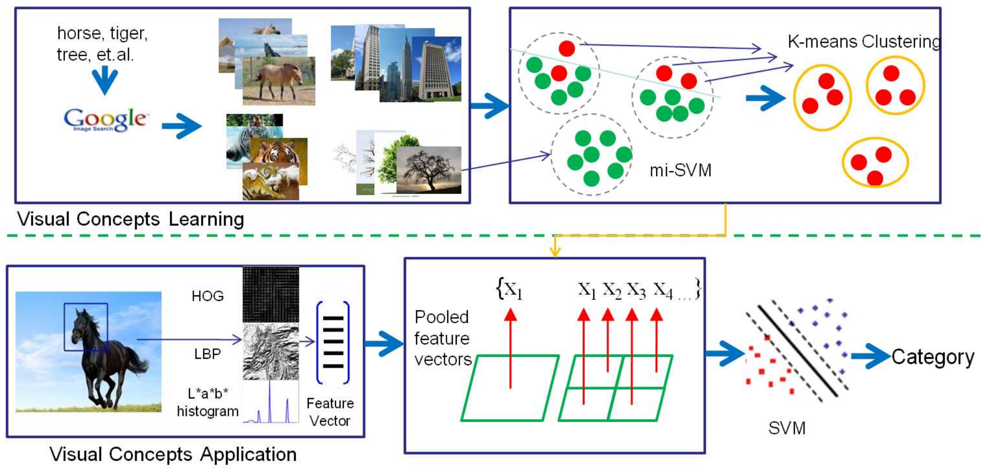


Figure 4.1: The flow chart of learning visual concepts from words.

4.2.1 Word Selection and Image Collection

The literal words are selected from ImageNet [DDS09], which is based on WordNet [MIL95] and Classeme [TSF10]. For the words with similar meanings, *e.g.*, "people", "guest", "worker", and "judge", we keep the most generic one. In all, $M = 716$ words are selected. Most of the words are representative ones of the popular categories in ImageNet such as "animal", "plants", "scenes", "activities", "foods", and "materials". For each word, we crawled the top 400 images from google.com and the top 30 images from bing.com and merged the images by removing the duplicates. For each category (word), around 400 images are retained.

Fig. 4.2 shows the top ranked images for 26 words. From Fig. 4.2, we can see

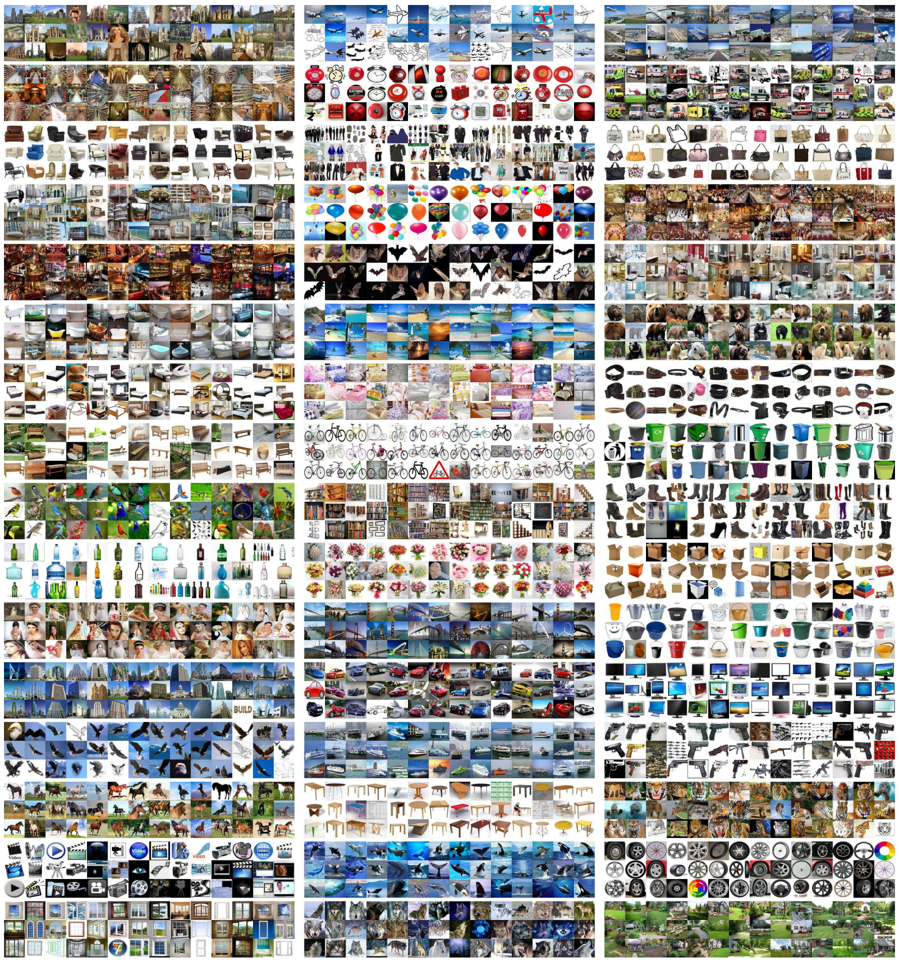


Figure 4.2: Sample images collected for 48 words.

that most of the retrieved images are generally of high relevance to the query word. Also, these images provide sufficient diversity stemming from the intra-category variances. For example, for the word “table”, besides the images of dinning tables, images of spreadsheets appear as well. The retrieved images for words such as “video” and “bird” are even more diverse. The diversity in these crawled images makes it inappropriate to train only a single classifier on the images, forcing us to investigate the multiple cluster property. Further, the object of interest usually does not occupy the entire image, making the multiple instance learning formulation a natural fit for this task.

4.2.2 Saliency Guided Bag Construction

The problem of visual concept learning is firstly unsupervised because we did not manually label or annotate the crawled images. However, if we view the query words as the labels for the images, the problem can be formulated in a weakly supervised setting, making our problem more focused and easier to be tackled.

Firstly, we convert each image to a bag of image patches with size greater than or equal to 64×64 that are more likely to carry semantic meanings. Instead of having randomly or densely sampled patches as th in [SGE12], we adopt a saliency detection technique to reduce the search space. Saliency detection assumes that the object of interest is generally salient in an image. Fig. 4.3 shows sample saliency detection results (the top 5 saliency windows for each image) by [FWT11], a window based saliency detection method. From Fig. 4.3, we observe that within the top 5 saliency windows, objects such as airplanes, birds, caterpillars, crosses, dogs, and horses are covered by the saliency windows. In addition, for the airplane and the caterpillar, the salient windows naturally correspond to the parts. This illustrates the benefit of the use of saliency detection: it helps to identify the regions and parts with more significance naturally. In our experiment, the top 50 salient windows are used as the instances of a positive bag directly. For large salient windows with sizes greater than 192×192, smaller patches within them are sampled, resulting in possible parts of the relevant patterns.

Although the saliency assumption is reasonable, not all category images satisfy this assumption. For example, for the images of "beach", the salient windows only cover patterns such as birds, trees, and clouds (see the salient windows of the "beach" image in Fig. 4.3). Although these covered patterns are also related to "beach", they cannot capture the scene as a whole because an image of "beach" is a mixture of visual concepts including sea, sky, and sands. To avoid missing non-salient regions for a word, besides using the salient windows, we also randomly sample some image patches from non-salient regions. As non-salient regions are often relatively uniform with less variation in the appearance, a smaller number of patches are sampled from the regions. After the patches are sampled, we perform overlap checks between the image patches with similar scale is performed. If two patches are of the similar scale and have high overlap, one patch will be removed.

Each bag constructed in this way thus consists of patches from both salient and non-salient regions. A portion of the patches may be unrelated to the word of interest, e.g., the patches corresponding to the sea in the image of "horse" in Fig. 4.3. Such patches are uncommon for the word "horse", and will be naturally filtered under the multiple instance learning framework.

Thus, for a word with N ($N \approx 400$) images, N bags $\{B_i, 1 \leq i \leq N\}$ can be constructed, each bag $B_i = \{\mathbf{x}_{ij}, 1 \leq j \leq m\}$, where m is the number of patches sampled for the image and is about 150 in our work; $\mathbf{x}_{ij}$ is the descriptor of the patch.

4.2.3 Our Formulation

To learn visual concepts from the bags constructed above, there are two basic requirements: 1) the irrelevant image patches should be filtered, and 2) the multiple cluster property of these visual patches should be investigated. Many methods meet these two requirements. In this work, we simply use the max-margin framework for multiple instance learning (miSVM) in [ATH02] to learn a linear SVM for each word, and then perform clustering on the positive instances labeled by the linear SVM. It is worth mentioning that another formulation for learning the multiple instance multi-classes problem can also be used, but it is not the main focus of this work.

In multiple instance learning, the labeling information is significantly weakened as the labels are assigned only to the bags with latent instance level lables. In

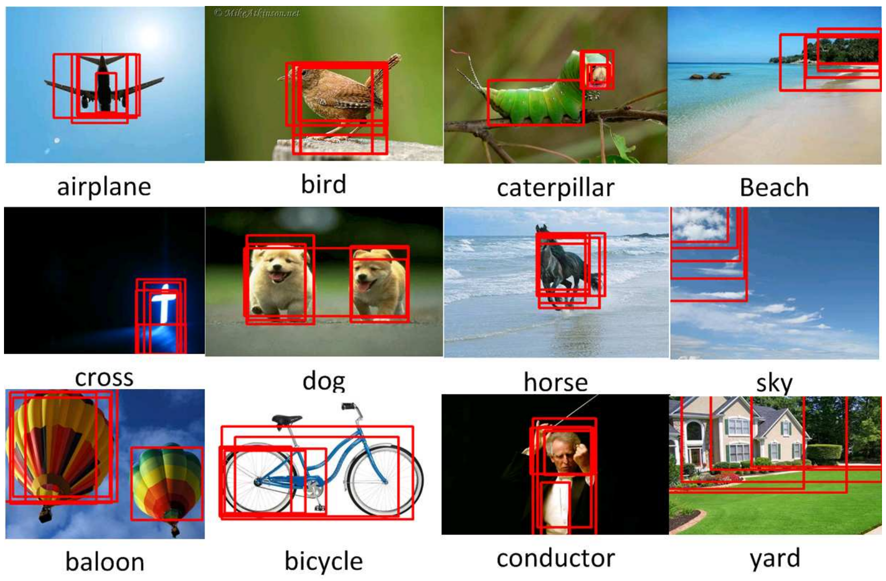


Figure 4.3: Top five salient windows for images from 12 words. Except for the words "sky", "beach", and "yard", the patterns of interest can be covered by a few top salient windows. For objects such as "caterpillar", "bicycle", and "conductor", parts can be captured by the salient windows.

[ATH02], the relationship between the bag level labels and the instance level labels is formulated as a set of linear constraints. With these linear constraints, soft-margin SVM is formulated into a mixed integer programming problem, which can by solved heuristically by iterating two steps: 1) given the instance level label y for an instance $\mathbf{x}$, solving the optimization discriminant function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ via Quadratic programming, where $\mathbf{w}$ is the weight vector, and b is the bias term and 2) given the discriminant function f, updating the instance level labels y. For more details on miSVM, the readers can refer to [ATH02].

4.2.3.1 Visual Concept Learning via miSVM

Using miSVM and assigning the literal words as the labels for the Internet images, visual concept learning for each word can be converted from an unsupervised learning problem into a weakly supervised learning problem. For a word k, its bag B_i is assigned with a label $Y_i = 1$. The instance level label y_{ij} for each instance $\mathbf{x}_{ij} \in B_i$ is unknown and will be automatically discovered by miSVM. For negative bags, we create a large negative bag B^- using a large amount of instances (patches) from words other than the word of interest. The number of instances in B^- is generally $5 \sim 10$ times more than the number of all the instances in the positive bags. The purpose of creating the large negative bag is to model the visual world, making the visual concepts learned for a word discriminant enough from the other words. For example, for words such as "horse" and "cow", using a large negative bag B^-, the common backgrounds such as the grassland and the sky can be filtered.

Based on $\{B_i, 1 \le i \le N\}$ and B^-, a linear SVM f^k can be learned by miSVM for the k-th word. The positive patches related to the word are also automatically found by miSVM. Given a patch, the linear SVM f^k can output a confidence value indicating the relevance of the patch to the word of interest. Therefore, the linear SVM f^k itself can be treated as a visual concept that models the patches of a word as a whole. We call it a *single-concept classifier*. Due to the embedded multi-cluster nature of diversity in the image concepts, a single classifier is insufficient to capture the diverse visual representations to a word concept. Thus, we apply another step in our algorithm: the positive instances (patches) automatically identified by the single-concept classifier are clustered to form some codes $C^k = \{C_1^k, C_2^k, ..., C_n^k\}$. We call these codes multi-cluster visual concepts. Different from the single-concept classifier, each multi-cluster visual concept corresponds to a compact image concept.

Therefore, for each word k, we learn two types of visual concepts, the single-concept classifier and the multi-cluster visual concepts C^k. From Internet images of the M words, we can learn M single-concept classifiers $F = \{f^1, ..., f^M\}$, and a set of multi-cluster visual concepts $C = \{C^k, 1 \le k \le M\}$. The single-concept classifiers and the visual concepts can be applied to novel images as the descriptors for categorization.

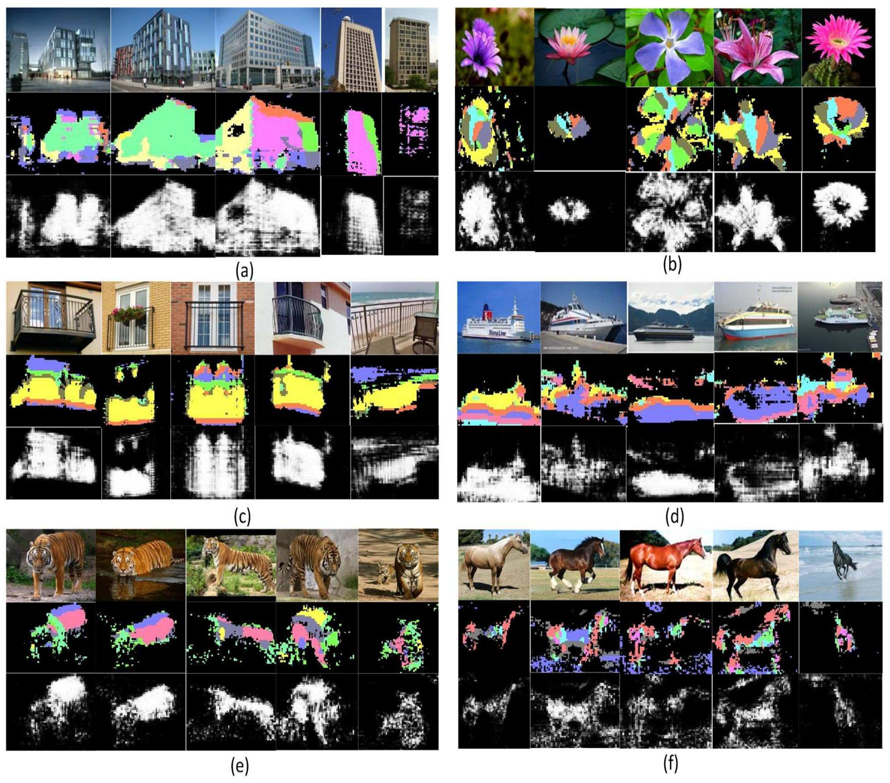


Figure 4.4: Illustration of the single-concept classifiers and the multi-cluster visual concepts for 6 words. (a) “building”, (b) “flower”, (c) “balcony”, (d) “ferry”, (e) “tiger”, and (f) “horse”. For each word, the first row shows the original images; the second row shows the assignment of the codes, and the third row shows the outputs of the single-concept classifier (the linear SVM for each word). See text for details.

In Fig. 4.4, we illustrate the outputs of the single-concept classifiers on the images, as well as the assignments of patches to the multi-cluster visual concepts. For clarity, for each word we cluster six multi-cluster visual concepts from the positive patches and assign them different colors randomly. For each image, we sample dense mid-level patches and apply the single-concept classifier to label the patches. We then assign the patches labeled as positive to the six multi-cluster visual concepts in a nearest neighborhood manner and display the colors of the assigned visual concepts in the centers of the patches.

The third row in Fig. 4.4 shows the outputs of the single-concept classifiers. Though learned in a weakly supervised manner, the single-concept classifiers can predict rather well. However, it cannot capture the diverse patterns of the patches, e.g., for the word "building", the walls of the left two images are different from the walls of the right three images. On the contrary, the multi-cluster visual concepts can capture such differences. The walls in the left two images of the word "building" have the same patten, and they are assigned to the same multi-cluster visual concept that has relatively sparse and rectangle windows (indicated in green). The walls on the right three images have a different pattern and they are assigned to another visual concept that has square and denser windows (indicated in magenta). For the word "balcony", the columns are assigned to a multi-cluster visual concept indicated in yellow. For the other four "words", the objects of interest are generally a combination of several multi-cluster visual concepts. This illustrates that the single-concept classifiers and the multi-cluster visual concepts correspond to different aspects of images and complement each other.

4.2.4 Application for Image Classification

As our visual concept representation has two components, the single-concept classifiers $F = \{f^1, ..., f^M\}$ and the multi-cluster visual concepts $C = \{C^k, 1 \leq k \leq M\}$, we apply the two components separately on novel images. Each novel image is divided into grids of a three-level spatial pyramid [LSP06b]. The single-concept classifier f^k is applied to the densely sampled patches from the grids, and the responses of the classifiers are pooled in a max-pooling manner. For natural images, the objects are generally varying in different scales, and we run the classifiers on the novel images on these scales. Since our method works on the patch level and the visual concepts are learned with image patches of different scales, two or three scales are enough for testing images. The pooled responses across different scales are concatenated, leading to a feature vector with dimension $M \times 21 \times$ number of scales.

We use the multi-cluster visual concepts $C = \{C^k, 1 \leq k \leq M\}$ in a simple way as a single codebook in a spatial pyramid matching manner (SPM) [LSP06b]: multi-scale mid-level patches are assigned to the multi-cluster visual concepts via hard assignment; a histogram is constructed for each grid of the three-level spatial pyramid and the feature is the concatenated version of the histograms of all the multi-cluster visual concepts. In this way, for each novel image, we obtain a feature vector of dimension $M \times n \times 21$, where n is the number of multi-cluster visual concepts for each word.

Definitely, there are several other options. One is to train a linear classifier model for each visual concept, and apply the classifiers to the novel images. In this work, we simply use the basic scheme to illustrate the effectiveness of the visual concepts we learned.

Finally, the features corresponding to the single-concept classifiers and the multi-cluster visual concepts are combined like multiple kernel learning [BLJ04, LCB04]. The kernels K_F for the single-concept classifiers and K_C for the multi-cluster visual concepts are computed respectively and combined linearly: $K = wK_F + (1 - w)K_C$. In our work, as there are only two kernels, instead of using advanced techniques such as the SMO algorithm in [BLJ04], we can simply use cross-validation to determine the best w^*. χ^2 kernel is used in the experiments, and it can be computed efficiently using the explicit feature map in [VZ12, VF08].

4.3 Experiments and Results

On the PASCAL VOC 2007 [EVW], scene-15 [LSP06b], MIT indoor scene [QT09], UIUC-Sport [LF07] and Inria horse [FJS07b] image sets, we evaluate the visual concepts learned from the Internet images. On these image sets, the visual concepts achieve the state-of-the-art performances, demonstrating its good cross-dataset generalization capability. Also, as a type of generic knowledge from Internet images, when used with the specific models learned from specific image sets, the results can be further improved to a large extent.

4.3.1 Implementations

For each patch, we use HOG [DT05] (of 2048 dimensions), LBP [ZP07] (of 256 dimensions) and the $L^*a^*b^*$ histogram (of 96 dimensions) as the feature; these features are concatenated, leading to a feature vector of dimension 2400. As from Figure 4.5, by combining the three kinds of features, better result of the single-concept classifiers can be achieved.

The toolbox of LIBLINEAR [FCH08b] is adopted for efficient training; for each word, five iterations are used in miSVM. To create the visual concepts, on the patches labeled as positive by miSVM, 20 clusters are found using K-means; Thus, $716 \times 20 = 14320$ multi-cluster visual concepts are created for the 716 words.

We have created another two codebooks of size 14320. The first codebook is created by quantizing the densely sampled multi-scale image patches from images of all the words. The second codebook is created by finding 20 clusters from the images for each word. In the following, we name the first codebook KMS-ALL, and the second codebook KMS-SUB. As the two codebooks are created without using the saliency assumption and the multiple instance learning framework, they serve as two good baselines.

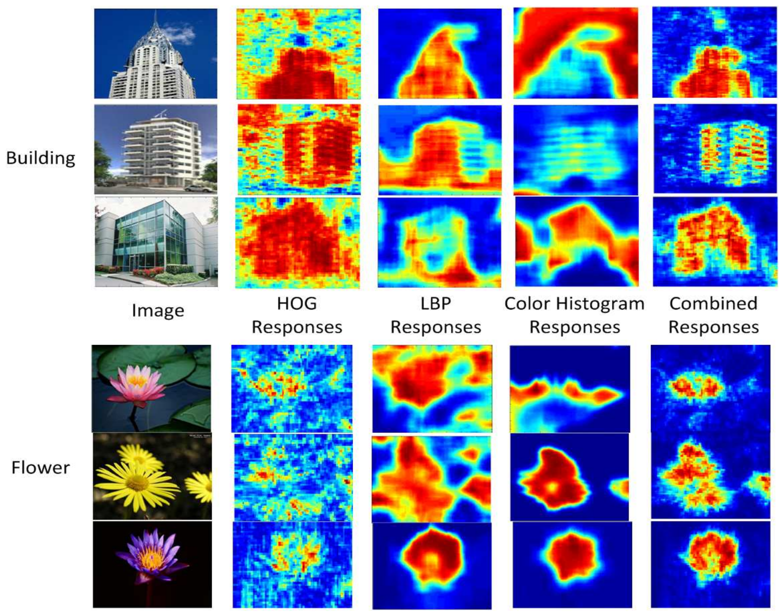


Figure 4.5: Illustration of feature combination: better response can be achieved by combining HoG, LBP, and L*a*b* histogram.

4.3.2 Quantitative Results

PASCAL VOC 2007 Image Set This dataset contains 20 object classes and 9963 images. It is split into training, validating and testing sets, and the mean average precision (mAP) of the 20 categories on the testing set is reported. The dataset is challenging, with large intra-class variances, cluttered backgrounds, and scale changes. When applying the visual concepts to the dataset, image patches of three scales 64×64, 128×128 and 192×192 are used.

In Table 4.1, we compare the mAPs. Firstly, we compare the visual concepts with the two baselines KMS-ALL and KMS-SUB. The multi-cluster visual concepts outperform both KMS-ALL and KMS-SUB, indicating that, the multi-cluster visual concepts learned are more effective. Though there are only 716 single-concept classifiers, they perform reasonably well, achieving an mAP 51%. By combining the single-concept classifiers and the multi-cluster visual concepts, the mAP is 57.5%, much higher than that of KMS-ALL and KMS-SUB.

We also compare our visual concepts with the improved Fisher-kernel (FK), locality-constrained linear coding (LLC)[WYY10b], and vector quantization (VQ). The fisher kernel starts from a Gaussian Mixture-Model (GMM), and concatenates the average first and second order differences between the patch descriptors and the centers of the GMM, leading to a feature vector of very high dimension. In [PSM10], the Fisher-kernel is improved by reducing the dimensionality of the patch descriptors using PCA. LLC [WYY10b] projects the patch descriptors to the local linear subspaces spanned by some visual words closest to the patch descriptors, and the feature vector is obtained by max-pooling the reconstruction weights. The improved Fisher-Kernel and LLC stand for the state-of-the-arts. For FK, LLC and VQ, the results reported here are from the image classification toolbox in [CLV11]. In [CLV11], multi-scale dense SIFT descriptors are used as the local features and the χ^2 kernel is used in SVM when classifying the images. From Table 4.1, we can observe that even though we do not use images from PASCAL VOC 2007 in the learning stage, the result of our visual concepts approach is comparable to that of the states-of-the-arts.

We investigate the complementariness of our visual concepts with the model learned from the images of the PASCAL VOC 2007 image set with advanced Fisher-kernels. The kernel matrices of the visual concepts and the improved Fisher-Kernels are combined linearly. The combination weight is learned on the validating set. For the improved Fisher-kernel, its result reported in [CLV11] is 61.69%, but when we run the toolbox with the suggested experimental settings, we get the mAP 59.6%. By combining the improved Fisher-kernel and our visual concepts, the result is boosted to 62.9%. This illustrates that our visual concepts do add extra information useful to the models learned from specific data sets.

Multiple clustered instance learning (MCIL) [XZC12] investigates the multiple cluster property at the instance level in the MIL-Boost framework. We applied M-

FK	LLC-25k	VQ-25K	MCIL	KMS-SUB
59.6%	57.66%	56.07%	43.3%	53.9%
KMS-ALL	SCCs	MVC	VC	FK+VC
53.3	51%	55.6%	57.5%	62.9%

Table 4.1: The mean average precisions on PASCAL VOC 2007 image set. FK: the improved Fisher-kernel with 256 components; LLC-25k: LLC with $25,000$ codes; VQ-25k: vector quantization with $25,000$ codes; MCIL: multiple clustered instance learning[XZC12]; KMS-SUB: the codebook created by clustering on each word; KMS-ALL: the codebook created by clustering on the image data from all the words; SCCs: the single-concept classifiers; MVC: the multi-cluster visual concepts; VC: the visual concepts, combination of the single-concept classifiers and multi-cluster visual concepts; FK+VC: combining the improved fisher kernel with our visual concepts.

CIL to learn a mixture of 20 cluster classifiers for each word, and used the outputs of the cluster classifiers as the features to encode the novel images. The result of MCIL is much worse than that of the visual concepts. The reason is that, in M-CIL, as the number of weak classifiers increases, the number of positive instances decreases dramatically and the cluster classifiers in MCIL learn little knowledge about the image set because of the lack of positive instances. Also, there is no competition between the cluster classifiers in MCIL, making the multiple cluster property of the image data not fully investigated.

Scene Classification We evaluate the visual concepts in the task of scene classification on three scene image sets, Scene-15 [LSP06b], MIT indoor scene [QT09], and UIUC-Sport event [LSX10]. Scene-15 has 15 natural scene classes; 100 images from each class are randomly selected for training and the remaining images are used for testing. UIUC-Sport has 8 complex event classes; 70 images from each class are randomly sampled for training and 60 images are sampled for testing. On both Scene-15 and UIUC-Sport, we run the experiments for 10 rounds, and report the average classification accuracy. MIT Indoor scene consists of 67 clustered indoor scene categories and has fixed training/testing splits.

On the scene image sets, image patches of two scales 64×64, 128×128 are used. The results are reported in Table 4.2. On the three datasets, our visual concepts approach outperforms KMS-ALL and KMS-SUB significantly. Object bank learns detection models for 200 objects from supervised data. Even though our visual concepts are learned in a weakly supervised manner, the visual concepts still outperform the detection models of object bank. The main reason for the superiority of our visual concepts is that, while object bank tries to capture an object using a single detection model, our method can capture the multiple cluster property with $14,200$ visual concepts and can model the diversity of the Internet images. We also test vector quantization (VQ) with $10,000$ codes on the three image sets using the toolbox [CLV11]. With such a large amount of codes, VQ performs surprisingly well on the UIUC-Sport and the MIT Indoor scene sets. On all the three scene image sets, our visual concepts perform comparably to VQ though we do not use the images from those image sets. By combining the VQ with our visual concepts, the performance can be boosted significantly. Relatively, the improvement is about 3% on the Scene-15 and UIUC-Sport image sets, and 10% on the MIT indoor scene set. For VQ, with the number of codes increased, the performance will saturate: we have tested VQ with $24,000$ codes on the MIT indoor scene image set, and the accuracy is 47.1%, even a slight decrease. From Table 4.2, we can see that, our method also outperforms recent methods such as [WLJ12], [NHG12],[YYG09b], [SGE12] and [KVR12].

Inria Horse Image Set INRIA horse dataset contains 170 horse images and 170 background images taken from the Internet. We randomly selected half of the images for training and the remaining images for testing and run the experiments

	Scene-15	UIUC-Sport	MIT-Indoor
Object Bank [LSX10]	80.9%	76.3%	37.6%
Yang *et al.* [YYG09b]	80.4%	-	-
Li *et al.* [LF07]	-	73.4%	-
Singh *et al.* [SGE12]	-	-	38%
Pandey *et al.* [PL11]	-	-	43.1%
Quattoni *et al.* [QT09]	-	-	26%
Niu *et al.* [NHG12]	78%	82.5%	-
Wang *et al.* [WLJ12]	80.43%	-	33.7%
Kwitt *et al.* [KVR12]	82.3%	83.0%	44.0%
KMS-ALL	78.7%	81.5%	38.8%
KMS-SUB	80.4%	83.2%	41.9%
VQ	82.1%	85.6%	47.6%
VC	83.4%	84.8%	46.4%
VC+VQ	**85.4%**	**88.4%**	**52.3%**

Table 4.2: The classification accuracies on the scene datasets.

for 10 rounds. On this image set, the accuracy of our visual concepts is 92.47%, better than the accuracy 91.4% of VQ with $10,000$ codes and 85.3% in [MNJ08b].

4.4 Conclusion

In this work, we have introduced a scheme to automatically exploit mid-level representations, called visual concepts, from large-scale Internet images retrieved using word-based queries. From more than a quarter of a million images, over 14,000 visual concepts are automatically learned. These learned visual concepts are generic and have good cross-dataset generalization capability; when combined with the models learned from specific dataset, our algorithm improves the state-of-the-arts to a large extent, demonstrating the complementariness between the visual concepts and the image content in specific datasets.

CHAPTER 5

Disagreement-Based Multi-System Tracking

5.1 Introduction

Object tracking has been a long standing problem in vision. Once a tracker gets initialized, it starts to track the target in a video by performing two steps: (1) making a prediction about the location of the target, and (2) updating its object model (location, appearance, and shape) based on the prediction. This is in spirit very similar to the bootstrapping and learning procedure in a learning algorithm. With the recent success in detection-based tracking approaches, an increasing amount of work has treated the tracking problem as a semi-supervised learning problem [AVI05, TBZ07, GLB08, LRL08, BYB09]. Picking a target to track at the beginning provides supervised data; the remaining of the frames for the tracker to explore do not contain label information and thus is unsupervised. Due to the errors introduced in both the prediction and model updating stage, nearly any tracker will eventually fail with the errors being accumulated over the time.

Disagreement-based semi-supervised learning approaches [ZL10], such as co-training or tri-training [BM98, ZL05], provide a mechanism to allow classifiers trained on different views or data samples to exploit unlabeled data. The learning process is a type of ensemble learning [DIE00, KUN02, BDM02]. It involves multiple classifiers which label the unlabeled data to update and improve each other [WZ07]. From a different angle, the use of multiple classifiers can be viewed as a fusion problem and it has been shown that fusing complementary features in a tracking system often leads to enhanced performances [WH04, SM02, TN01]. However, less efforts have been made in learning to fuse well-developed existing algorithms through semi-supervised learning; we will see later (in both theory and experiments) that a disagreement-based fusion significantly improves the performance over direct combination of features/systems [SM02, SS03, KL10].

In this disagreement-based multi-system tracking approach, we seek a balance between the current tracker and the level of agreements among other trackers. Our intuition is to find the location where the current tracker is confident but disagrees with other trackers, while other trackers reach a high degree of agreement. We provide both theoretical and experimental evidence to our approach and show much improved results over the state-of-the-art techniques on benchmark videos.

5.2 Related Work

A number of tracking methods have been proposed to perform fusion[WH04, SM02, I L06, ZYC10, SS03, TN01, KL10]. Different from [WH04, KL10] where multiple parts were tracked and correlated, we deal with a single target. In [SM02, SS03] multiple trackers were fused but these trackers represent different features and they were directly combined. In [I L06] the tracking approach was combined via the weighted combination of the PDFs. Different from [I L06], our method does not perform direct multiplication but seeks a balance between the PDF of one tracker and the degree of agreement by the other trackers; also, in our method, each tracker performs prediction separately maintaining certain independence and patches at the agreed positions can be recommended to update the other trackers. In [SWC09], the tracking combination method is trained for specific scenarios. Different from [SWC09], our method is based on the disagreement-based semi-supervised learning and do not require an off-line training process; also, it can be applied to general videos, and performs very well on a fairly large benchmark dataset. In [MC10], mutual information was used for the fusion. Here, the proposed fusion approach is based on the disagreements among the trackers. The most related work to our approach is [TBZ07], where the co-training idea was used to retrain classification-based trackers. However, [TBZ07] followed the standard co-training implementation using one specific type of classifier, SVM. In [ZYC10] several tracking algorithms were combined in a Bayesian framework whereas we here emphasize disagreement-based fusion through semi-supervised learning.

In disagreement-based semi-supervised learning, much of the work has been focused on using multi-view features [BM98] or different data samples [ZL05]. The spirit of all such kind of approaches [BM98, CS99, ZL05, LT08] is to train multiple classifiers with disagreements, and then label the unlabeled instances for each other to update/improve the model. [DLM99] provided PAC bounds with multi-view features, while [WZ07] provided a sufficient condition for multi-view as well as single-view features. Recently, a sufficient and necessary condition was proved for disagreement-based semi-supervised learning, by establishing a connection between disagreement-based and graph-based approaches [WZ10].

In this work, we emphasize taking advantages of having various well-developed tracking algorithms. In the democratic co-learning framework [ZG04], different algorithms are also used; however, their approach is for classification and a direct voting of all the methods is used. A main difference between tracking and classification is that there is no labeling information provided once the tracking process starts (it is a dynamic system), whereas most disagreement-based semi-supervised learning algorithms can still use labeled data in retraining. Notice that the existence of large disagreements among the classifiers is a premise for the learning or tracking process to continue [WZ07], while the prediction is made by seeking the agreements among the classifiers. For example, in [CS99, LT08], classifiers are learned so that they not only fit the supervised data well, but also themselves reach a high degree of agreement; the tri-training algorithm [ZL05] uses confident and agreed data from two classifiers to help the third classifier. Our agreement is used in the prediction stage like the bootstrapping stage in [ZG04], and we further emphasize the consistency with the information provided by the current tracker. Our work is also related to the active learning literature [DAS05] but we do not have humans in the loop.

5.3 Disagreement-Based Tracking

The problem of making predictions in a tracking system has its own unique characteristic, and directly applying the standard co-training formulation [TBZ07] may not necessarily yield a good solution. Instead, we take advantages of having well-developed existing algorithms, *experts*, and combine them by exploiting the disagreements among the experts. The differences in the intrinsic design of the existing systems will naturally lead to a certain amount of biases/variations, a property the disagreement-based approaches requires [WZ07].

5.3.1 Prediction of Single Tracker

In this section, we first clarify our notations for a single tracker. A tracker can be viewed as a learner denoted by $h^t = (A, f^t, X^t)$ since it always updates itself. Here, A is a specific tracking method e.g. mean-shift tracker [CRM00], or particle filtering [IB98]; f^t is the underlying appearance model about the target at time t which can be represented by a discriminative model [BYB09], generative model [LRL08], or template matching [CRM00]; X^t is the position of the target at time stamp t. Given a new image I^{t+1} at time stamp $t+1$, tracker h^t makes a prediction, X^{t+1}, about the position of the target and updates its underlying appearance model to f^{t+1}. We can view tracking a target of a tracker A as computing

$$\begin{aligned} q_A^{t+1}(x) \quad &\equiv \quad p_A(y_x = +1 | I^{t+1}(x), f^t) \cdot p(X^{t+1} = x, X^t) \\ \text{and} \quad & \sum_x q_A^{t+1}(x) = 1 \end{aligned} \tag{5.1}$$

Here $y_x = +1$ indicates the occurrence of target at location x and $I^{t+1}(x)$ is an image patch centered at x. Motion coherence is assumed that the prediction on the time stamp $t+1$ is smooth w.r.t. to the prediction on the time stamp t, for example, $p(X^{t+1} = x, X^t)$ can be a constant within a neighborhood of X_t and zero outside. This corresponds to the local search strategy adopted by most of the trackers.

Now that $q_A^{t+1}(x) \in [0,1]$ indicates how likely x_{t+1} is the correct position for the target. For an existing tracking algorithm, it may not strictly follow the formulation as in Eq. (5.1), but we still can use it so long as it outputs a probability map for the prediction.

5.3.2 Disagreement of Trackers

Suppose we have a set of existing trackers (experts) for making a prediction in a tracking system $S = \{h_i, i = 1..n\}$ with $n \geq 3$ being the number of trackers and each h_i is a tracker trained by tracking algorithm A_i. Given an input I^{t+1} at a time stamp $t+1$, each tracker h_i computes a $q_i^{t+1}(x)$ to make a prediction of random variable X. Let $p^{t+1}(x)$ denote the ground probability map which indicates how likely x is the correct position, our general objective is to combine the probability maps by different trackers to obtain high probability modes in the “ground-truth” $p^{t+1}(x)$.

A direct way to fuse the multiple trackers is by linearly combining the probability maps together [TN01]. Here, we call it direct tracker fusion (DTF), which serves as a baseline algorithm:

$$\bar{q}^{t+1}(x) = \frac{1}{n}\sum_{i=1}^{n} q_i^{t+1}(x) \tag{5.2}$$

with the hope that $\bar{q}^{t+1}(x) \rightarrow p^{t+1}(x)$ as each tracker being unbiased and independent. Algorithms like [TN01] perform in this way with an adaption in the weighting parameters. The target location is retrieved by $\check{x}^{t+1} = \arg\max_x \bar{q}^{t+1}(x)$. In DTF, at each time, all trackers use the same prediction, $\check{x}^{t+1}$, and each tracker updates its appearance model to f_i^{t+1} based on $\check{x}^{t+1}$ separately and continues the tracking process. Fusing trackers leads to improvement over the original ones (see Section 5.3.2.1 and Section 5.4 for theoretical and empirical justification respectively).

However, predicting the position for the target w.r.t. Eq. (5.2) has a big drawback, i.e., the average performance $\overline{q}^{t+1}(x)$ of the n trackers may be degenerated by one bad tracker in the group. Here we give an example to illustrate this: suppose there are four trackers f_1, f_2, f_3, f_4 and two candidate positions x^* and $x^{'}$ at $t+1$, where x^* is the correct position for the target. The outputs of the four trackers on the two candidate positions are $q_1(x^{'}) = 0.9$, $q_2(x^{'}) = 0.4$, $q_3(x^{'}) = 0.4$, $q_4(x^{'}) = 0.4$ and $q_1(x^*) = 0.1$, $q_2(x^*) = 0.6$, $q_3(x^*) = 0.6$, $q_4(x^*) = 0.6$. The tracker f_1 is very confident but disagrees with other trackers and makes a wrong prediction. To some extent, this kind of tracker which is confident but disagrees with other trackers can be thought of as a outlier tracker. If we fuse the four trackers with direct tracker fusion (DTF), the position $x^{'}$ will be predicted as the position for the target according to $x = \arg\max_x \overline{q}^{t+1}(x)$. Unfortunately, we get a wrong position $x^{'}$ due to the outlier tracker f_1 at $t+1$ although three trackers make correct prediction with confidence larger than 0.5.

Let ζ_{t+1} denote the probability mass on such an event that the average performance $\overline{q}^{t+1}(x)$ is degenerated by some outlier tracker f_i, i.e., the other $n-1$ trackers agree with each other and predict the correct position with high confidence while the DTF in Eq. (5.2) predict the position wrongly due to the outlier tracker f_i at $t+1$. Next, we give the formulation for combining the multiple trackers based on their disagreement to avoid this kind of event for the purpose of robustness. Given n trackers, we still let each tracker perform prediction separately. If the current tracker is confident but disagrees with other trackers while other trackers reach a high degree of agreement, the current tracker is prone to be drifted to the agreed position of other trackers to reach more robust predictions. Our intuition is that we seek a balance between the generated distribution $q_i^{t+1}(x)$ of the current tracker and the degree of agreement by the other trackers as

$$\begin{aligned} Q_i^{t+1}(x) &= (1-\alpha)q_i^{t+1}(x) + \frac{\alpha}{n-1}[\sum_{j=1,j\neq i}^{n} q_j^{t+1}(x)] \cdot \\ &\quad \delta(\forall j \neq i, q_j^{t+1}(x) \geq TH) \end{aligned} \tag{5.3}$$

and the specific location by the i-th tracker is $\tilde{x}_i^{t+1} = \arg\max_x Q_i^{t+1}(x)$. TH is a threshold corresponding to a confidence zone. α balances the importance of each tracker's own prediction and the influence from other trackers. The derivation of TH and α will be given in Section 5.3.2.1.

Note that the second term is non-zero only when all the other trackers have a high-degree agreement; this is different from the traditional fusion-based tracking [TN01] where weighted sum is performed; in addition, we emphasize that Eq. (5.3) focuses mainly on the places with high probability and it is not necessary to fit $p^{t+1}(x)$ at all xs as in the general statistical learning; our disagreement formulation in Eq. (5.3) can take advantage of this property.

Eq. (5.3) can be understood as the following: if the current tracker disagree with the other trackers while the other trackers are confident and agree with each other, the prediction of the current tracker will be influenced towards the agreed location (depending upon the overall probability map); otherwise, tracker h_i gives out a prediction as if there were no other trackers. In such a way, the trackers can keep relative independence and also enable *confident* interactions between each other. This makes our approach robust to outlier trackers. In addition, using the agreement of other trackers gives the overall system an ability to be self-aware of when the system starts to drift. This happens when all trackers have high entropy of $q_i^{t+1}(x)$ with large disagreement.

The overall output is then given by $x^{t+1\,*} = \arg\max_x \mathcal{Q}^{t+1}(x)$ and

$$\mathcal{Q}^{t+1}(x) = \frac{1}{n}\sum_{i=1}^{n} Q_i^{t+1}(x) \tag{5.4}$$

Note that $x^{t+1\,*}$ is the output of the overall system but it does not participate in the retraining of the individual trackers. The pseudo code of disagreement-based tracking is shown in Fig. 5.1. Tracking based on the disagreements among the trackers shows advantage over using a direct combination and we justify this point both theoretically and empirically in the following sections.

5.3.2.1 Theoretical Justification

We first show that a linear combination of multiple trackers as in Eq. (5.2), direct tracker fusion (DTF), gains improvement over the individual systems. Let $p^{t+1}(x)$ denote the ground truth which indicates how likely x is the correct position, and let $q_i^{t+1}(x) \in [0,1]$ be the output of algorithm A_i.

Lemma 3 *If we take an average of the predictions from all the experts: $\bar{q}^{t+1}(x) = \frac{1}{n}\sum_{i=1}^{n} q_i^{t+1}(x)$ as in Eq. (5.2), then the average is bounded in a PAC sense. We suppose that the n trackers are independent and unbiased: then $\overline{q}^{t+1}(x) \to p^{t+1}(x)$ as $n \to +\infty$.*

Proof For any small $\epsilon > 0$, with Hoeffding inequality, we get that $P(|\overline{q}^{t+1}(x) - p^{t+1}(x)| \geq \epsilon) \leq 2\exp(-2n\epsilon^2)$. ∎

Lemma 3 shows that $\overline{q}^{t+1}(x)$ can converge to the ground truth $p^{t+1}(x)$ exponentially. Let $error_i^{t+1}$ denote the error rate of the tracker f_i at $t+1$, i.e., the

```
Given $n$ trackers $\{h_i, i = 1..n\}$, each tracker $h_i = (A_i, f_i^t, X_i^t)$ adopts a specific tracking method $A_i$. At the time stamp $t = 0$, a target is manually identified located at $X^0$. All trackers start with the same $X^0$ and obtain their appearance model $f_i^0$. Given a new image $I^{t+1}$ at time stamp $t + 1$,

    • Each tracker $h_i$ searches a local neighborhood around $X_i^t$ and generate a probability map $q_i^{t+1}$ using Eq. (5.1).

    • Find modes of $\tilde{x}_i^{t+1}$ for $Q_i^{t+1}(x)$ as in Eq. (5.3). $\tilde{x}_i^{t+1}$ keeps a balance between the estimation of the current tracker and the level of agreements among other trackers.

    • Assign $X_i^{t+1} = \tilde{x}_i^{t+1}$, sample patches around $X_i^{t+1}$ and update the appearance model of each tracker to $f_i^{t+1}$ using the embedded model updating/learning rule in $A_i$

    • Based on $x^{t+1\,*} = \arg\max_x \sum_i Q_i^{t+1}(x)$, report the $x^{t+1\,*}$ as the tracking result for disagreement-based tracking (DBT).
```

Figure 5.1: Pseudo code of disagreement-based tracking.

probability that f_i predicts a wrong position for the target at $t + 1$, $error_{min}^{t+1} = \min_i\{error_i^{t+1}\}$ and $error_{max}^{t+1} = \max_i\{error_i^{t+1}\}$, we give the following theorem to show that fusing the multiple trackers according to Eq. (5.3) and Eq. (5.4) will improve the performance at least $\zeta_{t+1} - n(error_{max}^{t+1})^{n-1}$, contrasting to the direct tracker fusion (ζ_{t+1} was defined in the previous section).

Theorem 2 *If we fuse the multiple trackers according to Eq. (5.3) and Eq. (5.4) where $\alpha \geq \frac{2}{3}$ and $TH \geq \frac{1}{2}$, contrasting to the direct tracker fusion in Eq. (5.2), the performance at $t+1$ can be improved at least $\zeta_{t+1} - n(error_{max}^{t+1})^{n-1}$, where ζ_{t+1} is the probability of the event that the average performance $\overline{q}^{t+1}(x)$ is degenerated by some outlier tracker.*

Proof Let $\mathcal{X}^{t+1}$ denote the set of candidate positions at $t+1$. If there is some $x^* \in \mathcal{X}^{t+1}$, at which $q_i^{t+1}(x^*) \geq TH$ for all $i \in \{1, \ldots, n\}$, it is easy to find that such x^* is unique, since $TH \geq 0.5$ (Here we neglect the probability mass on the event that at $t{+}1$ there are two positions x and x' at which $q_i^{t+1}(x) = q_i^{t+1}(x') = \frac{1}{2}$). Considering Eq. (5.3) we get $\mathcal{Q}^{t+1}(x^*) = \frac{1}{n}\sum_{k=1}^{n} q_k^{t+1}(x^*)$, and x^* will be selected as the tracking result, no matter whether $\arg\max \mathcal{Q}^{t+1}(x)$ or $\arg\max \overline{q}^{t+1}$ is used. If for any $x \in \mathcal{X}^{t+1}$ there are less than $n-1$ trackers with $q_i^{t+1}(x) \geq TH$, then the second term of Eq. (5.3) is zero. So for any $x \in \mathcal{X}^{t+1}$, $\mathcal{Q}^{t+1}(x) = \frac{1-\alpha}{n}\sum_{k=1}^{n} q_k^{t+1}(x)$. Predicting the tracing result according to $\arg\max \mathcal{Q}^{t+1}(x)$ is equal to predicting according to $\arg\max \overline{q}^{t+1}$. Next we analyze the situation when there is some $\widehat{x} \in \mathcal{X}^{t+1}$, at which $q_i^{t+1}(\widehat{x}) < TH$ and $q_j^{t+1}(\widehat{x}) \geq TH$ for all $j \neq i$. Obviously, such $\widehat{x}$ is also unique, since $TH \geq 0.5$ and $n \geq 3$.

Case 1: $\widehat{x}$ is the correct position for the target at $t+1$. We will show that even if $q_i^{t+1}(\widehat{x})$ is very close to 0, i.e., tracker f_i is an outlier at $t+1$, it will not degenerate the fusion of the multiple trackers due to Eq. (5.3). We obtain

$$\begin{aligned} &Q_i^{t+1}(\widehat{x}) = (1-\alpha)q_i^{t+1}(\widehat{x}) + \\ &\frac{\alpha}{n-1}\sum_{j\neq i} q_j^{t+1}(\widehat{x}) \geq (1-\alpha)q_i^{t+1}(\widehat{x}) + \alpha\cdot TH \end{aligned} \tag{5.5}$$

$$Q_{j,j\neq i}^{t+1}(\widehat{x}) \;=\; (1-\alpha)q_j^{t+1}(\widehat{x}) \geq (1-\alpha)\cdot TH$$

Thus,

$$\begin{aligned} \sum_{k=1}^{n} Q_k^{t+1}(\widehat{x}) \geq \quad & (1-\alpha)q_i^{t+1}(\widehat{x}) + \alpha\cdot TH + \\ & (1-\alpha)(n-1)TH \end{aligned} \tag{5.6}$$

For an incorrect position $x' \neq \widehat{x}$, since $\sum_{x\in\mathcal{X}^{t+1}} q_k^{t+1}(x) = 1$, it is easy to see that

$$Q_i^{t+1}(x') \;=\; (1-\alpha)q_i^{t+1}(x') \leq (1-\alpha)(1-q_i^{t+1}(\widehat{x}))$$

$$\begin{aligned} Q_{j,j\neq i}^{t+1}(x') \;&=\; (1-\alpha)q_j^{t+1}(x') \leq (1-\alpha)(1-q_i^{t+1}(\widehat{x})) \\ &\leq\; (1-\alpha)(1-TH) \end{aligned} \tag{5.7}$$

Therefore,

$$\begin{aligned}\sum_{k=1}^{n} Q_k^{t+1}(x') \leq \quad & (1-\alpha)(1-q_i^{t+1}(\widehat{x})) \\ + \quad & (1-\alpha)(n-1)(1-TH) \end{aligned} \tag{5.8}$$

We see that in general $\sum_{k=1}^{n} Q_k^{t+1}(\widehat{x}) \geq \sum_{k=1}^{n} Q_k^{t+1}(x')$ for $\alpha \geq \frac{2}{3}$ and $TH \geq \frac{1}{2}$. This makes the correct position more robust, i.e., the prediction of the disagreement-based tracking will never be influenced even if f_i is a outlier tracker. So the improvement is at least ζ_{t+1}.

Case 2: $\widehat{x}$ is not the correct position for the target at $t+1$. Since $q_j^{t+1}(\widehat{x}) \geq TH$ for all $j \neq i$ and $TH \geq \frac{1}{2}$, $n-1$ trackers predict the wrong position $\widehat{x}$ as the tracking result at $t+1$. Now we bound the probability of such event. Since $error_j^{t+1} \leq error_{max}^{t+1}$ and the multiple trackers are assumed to be independent, the probability mass on the event that $n-1$ trackers predict the position mistakenly is at most $n(error_{max}^{t+1})^{n-1}$. The worst situation is that the fusion according to Eq. (5.3) performs worse than the direct tracker fusion in **case 2** completely. We get Theorem 2 proved. $\blacksquare$

From Theorem 2 we know that the fusion will get benefit from Eq. (5.3) under the situation that one bad tracker degenerates the direct tracker fusion. When n (the number of the trackers) is large, it would be difficult for the remaining $n-1$ trackers to achieve some agreement (See the experiment "Non-Relax" in Section 5.4). In practice, we can relax this constraint, e.g., when two or more trackers achieve agreement, the agreement term would take effect.

Note that the performance of Eq. (5.2) and Eq. (5.3) depends on the correlation between the trackers. The correlation depends on two factors: (1) the intrinsic design of the trackers; (2) the training samples used to train the trackers. Two trackers with the same design trained on the same set of samples are highly correlated, and two different types of trackers trained on the same set of samples are more correlated than those trained on different set of samples. If the n trackers are the same, then using Eq. (5.3) shows no advantage over Eq. (5.2).

In summary, Lemma 3 suggests that fusing the multiple experts directly might gain exponential improvement, contrasting to the single tracker; Theorem 2 shows that our disagreement-based fusion method can provide more robustness to the tracking system, which motivates the use of Eq. (5.3) by keeping a balance between the current expert f_i and the agreement from the other experts.

5.4 Experiments

In the experiments, we make a comprehensive comparison between the performance of disagreement-based tracking, direct tracker fusion, and the individual trackers. Four trackers are used and the experiment is conducted on 11 commonly tested videos (listed in Table 5.1). The trackers used are MilTracker [BYB09], the semi-supervised on-line boosting tracker (semiBoost)[GLB08], Incremental Visual Tracker (IVT)[LRL08], and Incremental Visual Tracker using edge information (IVTE).

Since the individual trackers perform prediction separately, the computational complexity of the proposed method only adds slight overhead over the individual ones with the multi-core processor and parallel computing. Compared with the large performance gain, this computational overhead is tolerable.

From the experiments, we observe that, statistically, each individual tracker gets significantly improved by using Eq. (5.3): the average center location error has been reduced by more than 12 pixels and the success rate has increased by 20% $\sim$ 40%. The result of DBT (Disagreement-Based Tracking) also outperforms the system by directly combining the original trackers, i.e., DTF, with 3.3 pixels reduction in center location error and 4.4% improvement in success rate. DBT also significantly outperforms PROST [SLS10].

5.4.1 Implementation details

While a wealthy body of tracking papers/systems have been reported, we found a few systems (with available source code) having decent performance on general videos. Here, we provide brief descriptions for these trackers we used with necessary changes made to them.

MilTracker adopts an online multiple instance learning algorithm to train a discriminative classifier. In order to handle the ambiguity of sampled patches, a bag of potentially positive image patches are extracted. MilTracker maintains a pool of Haar features and the online boosting mechanism is adopted.

SemiBoostTracker also adopts an online boosting mechanism and it formulates the update process in a semi-supervised fashion combined with a given prior. This helps to alleviate the drifting problem.

The IVT incrementally learns a low dimensional eigenspace representation to model the appearance changes of the object. In IVT model, the target is represented as a vector of gray-scale value, and the motion is modeled by an affine image warping. To propagate sample distributions over time, a particle filter framework is adopted. Since both MILTracker and SemiBoostTracker do not support affine transformation, we disabled the scaling and rotating ability of IVT. IVTE is similar to IVT, except that it uses level set as the feature.

The forms of the 4 tracking systems' outputs are rather different. MILTracker and SemiBoostTracker produce scores on local search regions; IVT and IVTE propagate probabilities via particles. In the experiment, we map the scores of MILTracker and SemiBoostTracker to the range $[0, 1]$ to produce probability maps q_{MIL} and q_{SBT} (The probability maps are normalized to make sure that $\sum_x q_A^{t+1}(x) = 1$). For IVT and IVTE, we keep the position entries $(a_i,\ b_i)$ of the particle and use a parzen window approach to estimate the probability for prediction. For a point $x = (a, b)$ on the image, its probability is calculated as

$q_{IVT}(x) = \sum_{i=1}^{M} w_i * \max\{0, 1 - \sqrt{(a-a_i)^2 + (b-b_i)^2}/L\}$. In our experiment, L is set to 15. A map q_{IVTE} is produced similarly as q_{IVT} for IVTE.

Based on q_{MIL}, q_{SBT}, q_{IVT}, and q_{IVTE} we respectively compute the corresponding Q_{MIL}, Q_{SBT}, Q_{IVT}, and Q_{IVTE} using Eq. (5.3) (Since 4 trackers are used, we relaxed Eq. (5.3) that when 2 trackers achieve confident agreement, the agreement term will take effect) and thus, each tracker makes its own prediction separately. As we have mentioned before, $\tilde{x}_i^{t+1}$ found by mean shift algorithm [CRM00] can represent multiple points (modes). For each tracker, e.g., MILTracker, the one mode with the maximum value is reported as its prediction, significant modes found are used to retrain the tracker and as the seeds for further search at the next time stamp. For the results reported in our experiment, α is set to be 0.67 as suggested by the theoretical section. The threshold TH in Eq. (5.3) is set as $0.8/(\Lambda/3)$ (Λ is the size of the search window). For each tracker, 2 modes are kept.

5.4.2 Quantitative Results

The average center location error The average center location error is a commonly used metric to measure the performance of tracking and is defined as the average error between the predicted locations to the ground truth. In Table 5.1, we summarize the results of the average center location error on all the 11 videos. It's clear that our disagreement based tracker outperforms the individual trackers, Direct tracker fusion, and the co-training scheme. For non-relax (using Eq. (5.3) directly without relaxation), it's less possible for the trackers to achieve some agreements, and the chance of interaction between trackers is reduced. Still the result is better than the individual trackers.

The success rate If the location error on one frame is less than a pre-specified threshold, the prediction is regarded as a successful prediction. The success rate is

Table 5.1: Comparison of Average Center Location Error. Non-Relax indicates to use Eq. (5.3) directly without relaxation;Co-Training stands for the results using co-training method.

videos	MilT	IVT	IVTE	SBT	DTF	Co-Training	Non-Relax	DBT
Girl	31.9	25.2	18.1	19.3	20.6	39.8	23.3	**13.4**
CokeCan	20.5	55.3	11.0	14.9	9.3	49.0	7.9	**6.6**
Tiger1	**15.9**	71.9	56.6	20.9	37.7	64.1	49.0	31.2
Sylv	10.9	44.0	19.5	16	19.5	31.7	**7.3**	10.8
StatOcc	27.8	3.3	4.8	74.4	**2.5**	41.2	26.2	3.0
David	22.9	4.9	16.9	26.4	7.0	9.2	7.9	**4.1**
Cliffbar	12.0	31.4	78.6	29.9	27.1	45.6	16.7	**8.5**
Surfer	9.2	6.7	23.9	67.6	5.1	**4.7**	5.1	4.8
faceocc2	20.1	14.2	9.1	17	6.5	12.4	12.0	**6.1**
Indoor	17.2	30.2	193.5	116.5	4.7	61.9	10.7	**4.5**
faceocc	27.1	11.8	11.3	**6.8**	8.9	23.3	**7.6**	9.7
In all	20.8	21.8	22.3	37.3	11.5	32.7	15.8	**8.2**

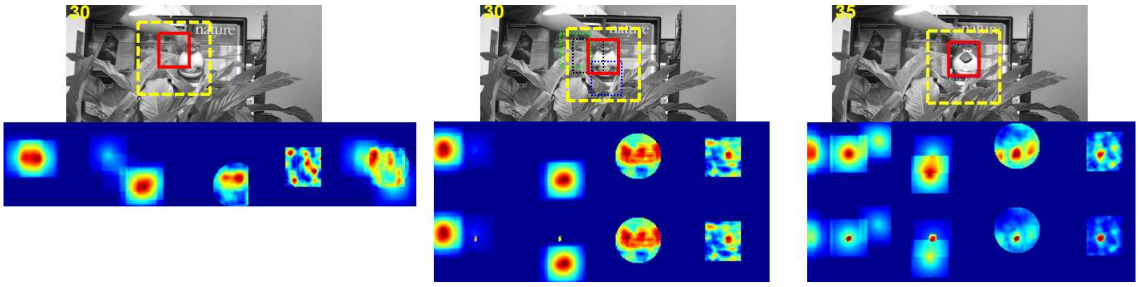
defined as the ratio of successful predictions over all the predictions. To compute the success rate, we set T as certain ratio of the average width of the target, i.e., $T = \beta * (w + h)/4$, where, w and h are the width and height of the target respectively. Conceptually, this is similar to the overlap score evaluation used in [SLS10] and thus, success rate is conceptually similar to the tracked percentile in [SLS10]. We observe that, the trackers can not track the target precisely at all times, if there is no overlap but the prediction of the tracker is not far from the target or within the search area of the tracker, it's often possible for the tracking process to recover. Our evaluation measurement can reflect such a phenomenon. We compare the success rate in Table 5.2 and from Table 5.2, the DBT achieves the highest success rate.

Table 5.2: Comparison of success rate when $\beta = 0.5$.

MilT	IVT	IVTE	SBT	DTF	Non-Relax	DBT
0.596	0.733	0.753	0.592	0.862	0.84	0.900

Comparison of probability maps In Fig. 5.2, we show how the probability maps are generated in Eq. (5.2) and Eq. (5.3) on a testing video, Tiger1. In Fig. 5.2, (a) shows the results by DTF. (b) and (c) display the probability maps generated by disagreement-based tracking. The probability maps inside the dashed yellow rectangle are shown below the screen shots. Underneath each figure, from left to right, the probability maps are IVT, IVTE, MILTracker, Semiboost tracker respectively (see the discussions about these trackers in the experiments). For DTF in (a), the fifth probability map is the combined map. For disagreement-based tracking in (b) and (c), the first rows shows the original probability maps, and the second row shows the Q_i^{t+1} computed by Eq. (5.3).

DTF drifts from the 30th frame; on this frame, the predictions of the trackers are rather diverse. In DTF, however, the individual tracker's prediction is not fully respected and it has to comply with the voted prediction; this is the primary reason for drifting and getting trapped; using Eq. (5.3) however leads to a more robust prediction. As we can see from the second figure, on the 30th frame, MILTracker and Semiboost tracker achieves certain agreement, but IVT and IVTE still complies with its own prediction since the agreement is weak. On the 35th frame, when MILTracker, IVT and Semiboost tracker achieve confident agreement, IVTE is pulled back to the agreed position and the four trackers merge again. The benefit of our disagreement-based tracking is obvious: the trackers then keep their relatively different traces and the risk of getting trapped is reduced.



(a) (b) (c)

Figure 5.2: Illustration of the probability maps where four trackers (experts) are adopted (the figures have been scaled for visualization).

Comparison with other methods PROST [SLS10] is another fusion based tracking algorithm that adopts 3 trackers (a template model, an optical-flow based mean-shift tracker and an online random forest tracker). Table 5.3 and Table 5.4 compare the average location errors and the tracked percentage (computed using the overlap score in [SLS10]) with PROST. From the two tables, we can observe that, our disagreement-based tracking outperforms PROST and achieves better tracked percentages on most of the videos.

The best experimental performance of Democratic Integration in [TN01] was achieved by using uniform qualities, which assigned equal weights to all the clues, and corresponded directly with DTF. In addition, we implemented the quality

Table 5.3: Comparison of average center location error with [SLS10]

noalign Method	Girl	tiger1	sylv	David	faceocc	faceocc2
[SLS10]	19.0	**7.2**	**10.6**	15.3	**7.0**	17.2
Ours	**13.4**	31.2	10.8	**4.1**	9.7	**6.1**

Table 5.4: Comparison of tracked percentage with [SLS10]

Method	Girl	tiger1	sylv	David	faceocc	faceocc2
[SLS10]	89	**79**	73	80	**100**	82
Ours	**97**	30	**83**	**100**	**100**	**100**

measure of normalized saliency and the performance was not as good as DBT: their average center location error is 13.8 with success rate 0.87.

We did not get the implementation of [TBZ07]. Nevertheless, we did experiment on some videos used in [TBZ07] and the results of DBT are better than [TBZ07] qualitatively (skipped here due to page limit). Moreover, we indeed implemented co-training and reported the result in Table 5.1 (average center location error 32.7), which is much worse than DBT.

Table 5.5: Performance by varying α and TH ($TH = R/(\Lambda/3)$)

R/α	0.8/0.3	0.8/0.67	0.8/0.85	0.7/0.67	0.9/0.67
Average Error	10.0	8.2	9.8	9.95	9.8
Success Rate	0.875	0.90	0.895	0.87	0.89

Robustness by varying the parameters In table 5.5, we summarize the performance of disagreement-based tracking by varying the parameters α and TH, which are the two key parameters in Eq. (5.3). We can see from this table, by varying TH and α, the results (especially the success rate) do not change too much. This demonstrates the robustness of disagreement-based tracking. The average center location error has relatively larger change because on portions of the videos Tiger1 and Indoor, disagreement-based tracking gets distracted to positions distant from the targets. In such cases, the success rate does not vary too much, but the center location error is increased.

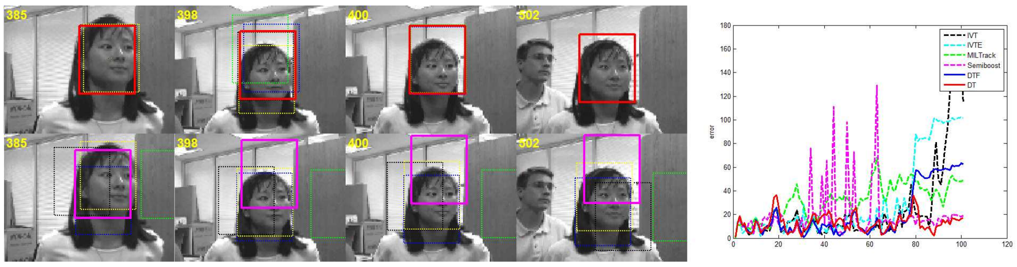


Figure 5.3: Comparison of tracking results on the video Girl. The first row: the results of disagreement-based tracking; the second row: the results of 4 individual trackers and direct tracker fusion; the right plot: the comparison of center location error; dotted green: IVT, dotted black: MILTracker, dotted blue: IVTE, dotted yellow: Semiboost tracker, solid Red: disagreement-based tracking, and solid magenta: Direct tracker fusion.

Screenshots of the results In Fig. 5.3, we compare the tracking results on the video Girl. This video undergoes several challenges: fast appearance change and occlusion. Although both MILTracker and IVTE can track the face of the girl successfully, the tracking process is not very stable. IVT drifts from the face at the 20th frame. From the 391th frame, direct tracker fusion also drifts and get trapped at the background of the images. As can be seen from both the screen shots and the error plot on the right of Fig. 5.3, we find that disagreement-based tracking tracks most robustly and accurately. In Fig. 5.3, we can also find a very nice property of democratic tracking that, the traces of the four trackers are similar but not exactly the same, thus, they can explore different spaces, recommend confident samples to other trackers, and thus avoid to be trapped at an incorrect position.

5.5 Conclusion

In this work, we have introduced a disagreement-based tracking method which fuses multiple existing tracking systems in the following way that seeks a balance between the coherence of the current tracker and the degree of agreements among other trackers. In such a way, it enables the interaction between trackers and keeps the appealing characteristics of the trackers at the same time. As illustrated in the experiments, the balance complies with the characteristic of tracking. Disagreement-based tracking can be built on top of various existing well-developed tracking systems utilizing their intrinsic biases. Adopting several state-of-the-art tracking algorithms, our approach is able to improve each of them by a large margin on widely used benchmark videos in the literature

References

[ASS83] P. ASSouad. "Plongements lipschitziens dans rn." *Bull. Soc. Math. France*, (4):429–448, 1983.

[ATH02] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. "Support Vector Machines for Multiple-Instance Learning." In *NIPS*, pp. 561–568, 2002.

[AVI05] S. AVIdan. "Ensemble Tracking." In *CVPR*, 2005.

[BAN22] Stefan BANach. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales." In *Fund. Math*, pp. 133–181, 1922.

[BDM02] Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin. "Exploiting Unlabeled Data in Ensemble Methods." In *ACM SIGKDD '02 Edmonton, Alberta CA*, 2002.

[BEN80] Jon Louis BENtley. "Multidimensional Divide-and-Conquer." *Commun. ACM*, **23**(4):214–229, 1980.

[BLJ04] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. "Multiple kernel learning, conic duality, and the SMO algorithm." In *ICML*, 2004.

[BM98] A. Blum and T. Mitchell. "Combining labeled and unlabeled data with co-training." In *COLT*, pp. 92–100, 1998.

[BN02] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation*, **15**:1373–1396, 2002.

[BRE96] Leo BREiman. "Bagging Predictors." *Machine Learning*, **24**(2):123–140, 1996.

[BRE01] Leo BREiman. "Random Forests." *Machine Learning*, **45**(1):5–32, 2001.

[BS09] Liefeng Bo and Cristian Sminchisescu. "Structured output-associative regression." In *CVPR*, pp. 2403–2410, 2009.

[BYB09] B. Babenko, M.-H. Yang, and S. Belongie. "Visual Tracking with Online Multiple Instance Learning." In *CVPR*, 2009.

[CDF98] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. "Learning to Extract Symbolic Knowledge from the World Wide Web." In *AAAI/IAAI*, pp. 509–516, 1998.

[CGP07] F. Perez Cruz, Z. Ghahramani, and M. Pontil. "Kernel conditional graphical models." *Predicting Structured Data*, pp. 265–282, 2007.

[CKY08] Rich Caruana, Nikolaos Karampatziakis, and Ainur Yessenalina. "An empirical evaluation of supervised learning in high dimensions." In *ICML*, pp. 96–103, 2008.

[CLV11] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. "The devil is in the details: an evaluation of recent feature encoding methods." In *British Machine Vision Conference*, 2011.

[CN06] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." In *ICML*, pp. 161–168, 2006.

[COL02] Michael COLlins. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms." In *EMNLP*, pp. 1–8, 2002.

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. "Real-Time Tracking of Non-Rigid Objects using Mean Shift." In *CVPR*, 2000.

[CS99] M. Collins and Y. Singer. "Unsupervised Models for Named Entity Classification." In *EMNLP*, 1999.

[CS01] Koby Crammer and Yoram Singer. "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines." *Journal of Machine Learning Research*, **2**:265–292, 2001.

[CT05] E. Candes and T. Tao. "Near-optimal signal recovery from random projections: universal encoding strategies." *IEEE Trans. Inform. Theory*, **52**(2):5406–5425, 2005.

[CZL07] Yuanhao Chen, Long Zhu, Chenxi Lin, Alan L. Yuille, and Hongjiang Zhang. "Rapid Inference on a Novel AND/OR graph for Object Detection, Segmentation and Parsing." In *NIPS*, 2007.

[DAS05] S. DASgupta. "Coarse sample complexity bounds for active learning." In *NIPS*, 2005.

[DDS09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database." In *CVPR09*, 2009.

[DF08] Sanjoy Dasgupta and Yoav Freund. "Random projection trees and low dimensional manifolds." In *STOC*, pp. 537–546, 2008.

[DIE00] Thomas G. DIEtterich. "Ensemble Methods in Machine Learning." In *INTERNATIONAL WORKSHOP ON MULTIPLE CLASSIFIER SYSTEMS*, pp. 1–15. Springer-Verlag, 2000.

[DLM99] S. Dasgupta, M. L. Littman, and D. McAllester. "PAC Generalization Bounds for Co-training." In *NIPS*, 1999.

[DLM09] Hal III Daumé, John Langford, and Daniel Marcu. "Search-based structured prediction." *Machine Learning*, **75**(3):297–325, 2009.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection." In *CVPR (1)*, pp. 886–893, 2005.

[EVW] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[EZ05] Mark Everingham and Andrew Zisserman. "The 2005 pascal visual object classes challenge." In *MLCW*, pp. 117–176, 2005.

[FCH08a] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. "LIBLINEAR: A Library for Large Linear Classification.", 2008.

[FCH08b] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. "LIBLINEAR: A Library for Large Linear Classification." *Journal of Machine Learning Research*, **9**:1871–1874, 2008.

[FDK07] Yoav Freund, Sanjoy Dasgupta, Mayank Kabra, and Nakul Verma. "Learning the structure of manifolds using random projections." In *NIPS*, volume 20, 2007.

[FEH09] Ali Farhadi, Ian Endres, Derek Hoiem, and David A. Forsyth. "Describing objects by their attributes." In *CVPR*, pp. 1778–1785, 2009.

[FGM10] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. "Object Detection with Discriminatively Trained Part-Based Models." *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(9):1627–1645, 2010.

[FH05] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Pictorial Structures for Object Recognition." *International Journal of Computer Vision*, **61**(1):55–79, 2005.

[FHH07] J. Friedma, T. Hastie, H. Hofling, and R. Tibshirani. “Pathwise Coordinate Optimization.” *The Annals of Applied Stat.*, 2007.

[FJ08] Thomas Finley and Thorsten Joachims. “Training structural SVMs when exact inference is intractable.” In *ICML*, pp. 304–311, 2008.

[FJS07a] V. Ferrari, F. Jurie, and C. Schmid. “Accurate Object Detection with Deformable Shape Models Learnt from Images.” In *CVPR*, 2007.

[FJS07b] Vittorio Ferrari, Frédéric Jurie, and Cordelia Schmid. “Accurate Object Detection with Deformable Shape Models Learnt from Images.” In *CVPR*, 2007.

[FS97] Y. Freund and R. E. Schapire. “A Decision-theoretic Generalization of On-line Learning And An Application to Boosting.” *J. of Comp. and Sys. Sci.*, **55**(1), 1997.

[FWT11] Jie Feng, Yichen Wei, Litian Tao, Chao Zhang, and Jian Sun. “Salient object detection by composition.” In *ICCV*, pp. 1028–1035, 2011.

[GG84] Stuart Geman and Donald Geman. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.” *IEEE Trans. Pattern Anal. Mach. Intell.*, **6**(6):721–741, 1984.

[GLB08] H. Grabner, C. Leistner, and H. Bischof. “Semi-Supervised On-line Boosting for Robust Tracking.” In *ECCV*, 2008.

[GTC10] Shenghua Gao, Ivor Wai hung Tsang, Liang tien Chia, and Peilin Zhao. “Local Features Are Not Lonely - Laplacian Sparse Coding for Image Classification.” In *CVPR*, 2010.

[HDF12] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. “Comparative Evaluation of Binary Features.” In *Proc. of ECCV*, 2012.

[HGS08] Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. “Cascaded Classification Models: Combining Models for Holistic Scene Understanding.” In *NIPS*, pp. 641–648, 2008.

[HSH13] Yi Hong, Zhangzhang Si, Wenze Hu, Song chun Zhu, and Ying nian Wu. “Unsupervised learning of compositional sparse code for natural image representation.” *Quarterly of Applied Mathematics*, 2013.

[I L06] E Rivlin I Leichter, M Lindenbaum. “A General Framework for Combining Visual TrackersThe ’Black Boxes’ Approach.” *It’l J. of Comp. Vis*, 2006.

[IB98] M. Isard and A. Blake. “CONDENSATION - Conditional Density Propagation for Visual Tracking.” *Int’l J. on Comp. Vis.*, 1998.

[JEW03] Pierre G. June, D. Ernst, and L Wehenkel. "Extremely Randomized Trees." In *Machine Learning*, volume 36, 2003.

[JUL75] BESag JULian. "Statistical Analysis of Non-Lattice Data." *Journal of the Royal Statistical Society. Series D (The Statistician)*, **24**(3):179–195, 1975.

[KL10] J. Kwon and K. M. Lee. "Visual Tracking Decomposition." In *CVPR*, 2010.

[KS07] S. Sathiya Keerthi and S. Sundararajan. "CRF versus SVM-struct for sequence labeling." In *Yahoo Research Technical Report*, 2007.

[KUN02] L. I. KUNcheva. "A theoretical study on six classifier fusion strategies." *IEEE Tran. on PAMI*, **24**(2):281–286, 2002.

[KVR12] Roland Kwitt, Nuno Vasconcelos, and Nikhil Rasiwasia. "Scene Recognition on the Semantic Manifold." In *ECCV (4)*, pp. 359–372, 2012.

[LBR07] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. "Efficient sparse coding algorithms." In *NIPS*, 2007.

[LCB04] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. "Learning the Kernel Matrix with Semidefinite Programming." *Journal of Machine Learning Research*, **5**:27–72, 2004.

[LF07] Li-Jia Li and Li Fei-Fei. "What, where and who? Classifying events by scene and object recognition." In *ICCV*, pp. 1–8, 2007.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In *ICML*, pp. 282–289, 2001.

[LO09] Y. Li and S Osher. "Coordinate Descent Optimization for ℓ^1 Minimization with Application to Compressed Sensing; a Greedy Algorithm." In *CAM Report*, 2009.

[LOW04] David G. LOWe. "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, **60**(2):91–110, 2004.

[LRL08] J. Lim, D. Ross, R.-S. Lin, and M.-H. Yang. "Incremental Learning for Visual Tracking." *Int'l J. of Comp. Vis.*, 2008.

[LRM12] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Jeffrey Dean, and Andrew Y. Ng. "Building high-level features using large scale unsupervised learning." In *ICML*, 2012.

[LSP06a] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In *CVPR*, 2006.

[LSP06b] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In *CVPR (2)*, pp. 2169–2178, 2006.

[LSX10] Li-Jia Li, Hao Su, Eric P. Xing, and Li Fei-Fei. "Object Bank: A High-Level Image Representation for Scene Classification & Semantic Feature Sparsification." In *NIPS*, pp. 1378–1386, 2010.

[LT08] B. Leskes and L. Torenvliet. "The value of agreement a new boosting algorithm." *J. of Comp. and Sys. Sci.*, 2008.

[MBPar] J. Mairal, F. Bach, and J Ponce. "Task-Driven Dictionary Learning." In *IEEE Trans. on PAMI*, To appear.

[MC10] J. L. Mundy and C. F. Chang. "Fusion of Intensity, Texture, and Color in Video Tracking Based on Mutual Information." In *AIPR*, 2010.

[MCC96] Andrew Kachites MCCallum. "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering." http://www.cs.cmu.edu/ mccallum/bow, 1996.

[MIL95] George A. MILler. "WordNet: A Lexical Database for English." *Commun. ACM*, **38**(11):39–41, 1995.

[MNJ08a] F. Moosmann, E. Nowak, and F. Jurie. "Randomized Clustering Forests for Image Classification." *IEEE Trans. on PAMI*, **30**(9):1632–1646, 2008.

[MNJ08b] Frank Moosmann, Eric Nowak, and Frédéric Jurie. "Randomized Clustering Forests for Image Classification." *IEEE Trans. Pattern Anal. Mach. Intell.*, **30**(9):1632–1646, 2008.

[NG07] Nam Nguyen and Yunsong Guo. "Comparisons of sequence labeling algorithms and extensions." In *ICML*, pp. 681–688, 2007.

[NHG12] Zhenxing Niu, Gang Hua, Xinbo Gao, and Qi Tian. "Context aware topic model for scene recognition." In *CVPR*, pp. 2743–2750, 2012.

[OPF06] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. "Generic Object Recognition with Boosting." *IEEE Trans. on PAMI*, **28**(3):416–431, 2006.

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. "A comparative study of texture measures with classification based on featured distributions." *Pattern Recognition*, **29**(1):51–59, 1996.

[OPM02] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns." *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**(7):971–987, 2002.

[PG11a] Devi Parikh and Kristen Grauman. "Interactively building a discriminative vocabulary of nameable attributes." In *CVPR*, pp. 1681–1688, 2011.

[PG11b] Devi Parikh and Kristen Grauman. "Relative attributes." In *ICCV*, pp. 503–510, 2011.

[PL11] Megha Pandey and Svetlana Lazebnik. "Scene recognition and weakly supervised object localization with deformable part-based models." In *ICCV*, pp. 1307–1314, 2011.

[PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. "Improving the Fisher Kernel for Large-Scale Image Classification." In *ECCV (4)*, pp. 143–156, 2010.

[QT09] Ariadna Quattoni and Antonio Torralba. "Recognizing indoor scenes." In *CVPR*, pp. 413–420, 2009.

[QUI86] J. R. QUInlan. "Induction of decision trees." *Machine Learning*, **1**, 1986.

[RAB89] Lawrence R. RABiner. "A tutorial on hidden markov models and selected applications in speech recognition." In *Proceedings of the IEEE*, pp. 257–286, 1989.

[ROM92] R. Reed, S. Oh, and R. J. Marks. "Regularization using jittered training data." In *IJCNN*, pp. 509–516, 1992.

[RS00] Sam Roweis and Laurence Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding." *Science*, **290**:2323–2326, 2000.

[SCH11] Mark SCHmidt. "UGM: Matlab code for undirected graphical models." 2011.

[SGE12] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. "Unsupervised Discovery of Mid-Level Discriminative Patches." In *ECCV (2)*, pp. 73–86, 2012.

[SJC08] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation." In *CVPR*, 2008.

[SLS10] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. "PROST: Parallel Robust Online Simple Tracking." In *CVPR*, 2010.

[SM02] N. Siebel and S. Maybank. "Fusion of multiple tracking algorithms for robust people tracking." In *ECCV*, 2002.

[SMJ10] David Sontag, Ofer Meshi, Tommi Jaakkola, and Amir Globerson. "More data means less inference: A pseudo-max approach to structured learning." In *NIPS*, pp. 2181–2189, 2010.

[SPY11] Zhangzhang Si, Mingtao Pei, Benjamin Yao, and Song-Chun Zhu. "Unsupervised learning of event AND-OR grammar and semantics from video." In *ICCV*, pp. 41–48, 2011.

[SS03] M. Spengler and B. Schiele. "Towards Robust Multi-cue Integration of Visual Tracking." In *MVA*, 2003.

[SWC09] B. Stenger, T. Woodley, and R. Cipolla. "Learning to Track with Multiple Observers." In *CVPR*, 2009.

[SZ12] Zhangzhang Si and Song-Chun Zhu. "Learning Hybrid Image Templates (HIT) by Information Projection." *IEEE Trans. Pattern Anal. Mach. Intell.*, **34**(7):1354–1367, 2012.

[TB10] Zhuowen Tu and Xiang Bai. "Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(10):1744–1757, 2010.

[TBZ07] F. Tang, S. Brennan, Q. Zhao, and H. Tao. "Co-Tracking Using Semi-Supervised Support Vector Machines." In *ICCV*, 2007.

[TGK03] Benjamin Taskar, Carlos Guestrin, and Daphne Koller. "Max-Margin Markov Networks." In *NIPS*, 2003.

[TIB96] R. TIBshirani. "Regression shrinkage and selection via the lasso." *J. Royal. Statist. Soc B.*, **56**(1):267–288, 1996.

[TJH05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. "Large Margin Methods for Structured and Interdependent Output Variables." *Journal of Machine Learning Research*, **6**:1453–1484, 2005.

[TN01] J. Triesch and C. von der Naksvyrg. “Democratic Integration: Self-organized Integration of Adaptive Visual Cues.” In *Neuroal Computation*, 2001.

[TRE02] The Penn TREEbank. “Penn’s linguistic data consortium. http://www.cis.upenn.edu/treebank.” 2002.

[TSF10] Lorenzo Torresani, Martin Szummer, and Andrew W. Fitzgibbon. “Efficient Object Category Recognition Using Classemes.” In *ECCV (1)*, pp. 776–789, 2010.

[TSL00] J. Tenenbaum, V.De Silva, and J. Langford. “A global geometric framework for nonlinear dimensionality reduction.” *Science*, **290**:2319–2323, 2000.

[TUR91] M TURk. “Eigenface for recognition.” *Journal of Cognitive Neuroscience*, 1991.

[UST09] T. Uno, M. Sugiyama, and K. Tsuda. “Efficient construction of neighborhood graphs by the multiple sorting method.” In *CoRR*, 2009.

[VF08] A. Vedaldi and B. Fulkerson. “VLFeat: An Open and Portable Library of Computer Vision Algorithms.”, 2008.

[VZ12] Andrea Vedaldi and Andrew Zisserman. “Efficient Additive Kernels via Explicit Feature Maps.” *IEEE Trans. Pattern Anal. Mach. Intell.*, **34**(3):480–492, 2012.

[WH04] Ying Wu and Thomas Huang. “Robust visual tracking by integrating multiple cues based on co-inference learning.” *In’l J. of Comp. Vis*, 2004.

[WLJ12] Liwei Wang, Yin Li, Jiaya Jia, Jian Sun, David P. Wipf, and James M. Rehg. “Learning sparse covariance patterns for natural scenes.” In *CVPR*, pp. 2767–2774, 2012.

[WOL92] D. WOLpert. “Stacked Generalization.” In *Neural Networks*, pp. 241–259, 1992.

[WW98] J. Weston and C. Watkins. “Multi-class support vector machines.” Technical Report, 1998.

[WYG09] John Wright, Allen Yang, Arvind Ganesh, Shankar Sastry, and Yi Ma. “Robust Face Recognition via Sparse Representation.” *IEEE Trans. on PAMI*, **31**(2), 2009.

[WYY10a] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. “Locality-constrained Linear Coding for image classification.” In *CVPR*, 2010.

[WYY10b] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. "Locality-constrained Linear Coding for image classification." In *CVPR*, pp. 3360–3367, 2010.

[WZ07] W. Wang and Z.-H. Zhou. "Analyzing co-training style algorithms." In *ECML*, 2007.

[WZ10] W. Wang and Z.-H. Zhou. "A new analysis of co-training." In *ICML*, pp. 1135–1142, Haifa, Israel, 2010.

[XZC12] Yan Xu, Jun-Yan Zhu, Eric I.-Chao Chang, and Zhuowen Tu. "Multiple clustered instance learning for histopathology cancer image classification, segmentation and clustering." In *CVPR*, pp. 964–971, 2012.

[YYG09a] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. "Linear spatial pyramid matching using sparse coding for image classification." *CVPR*, 2009.

[YYG09b] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas S. Huang. "Linear spatial pyramid matching using sparse coding for image classification." In *CVPR*, pp. 1794–1801, 2009.

[ZG04] Y. Zhou and S. Goldman. "Democratic Co-Learning." In *Inte'l Conf. on Tools with Art. Intell.*, 2004.

[ZH01] Ji Zhu and Trevor Hastie. "Kernel Logistic Regression and the Import Vector Machine." In *NIPS*, pp. 1081–1088, 2001.

[ZL05] Z.-H. Zhou and M. Li. "Tri-training: Exploiting unlabeled data using three classifiers." *IEEE Trans. on Know. and Data Eng.*, 2005.

[ZL10] Z.-H. Zhou and M. Li. "Semi-supervised learning by disagreement." *Knowledge and Information Systems*, **24**(3):415–439, 2010.

[ZP07] Guoying Zhao and Matti Pietikäinen. "Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions." *IEEE Trans. Pattern Anal. Mach. Intell.*, **29**(6):915–928, 2007.

[ZWW12] Jun-Yan Zhu, Jiajun Wu, Yichen Wei, Eric I.-Chao Chang, and Zhuowen Tu. "Unsupervised object class discovery via saliency-guided multiple class learning." In *CVPR*, pp. 3218–3225, 2012.

[ZYC10] B. Zhong, H. Yao, S. Chen, R.-R. Ji, X. Yuan, S. Liu, and W. Gao. "Visual Tracking via Weakly Supervised Learning from Multiple Imperfect Oracles." In *CVPR*, 2010.