Towards Scalable Analysis of Images and Videos

Bin Zhao

September 2014
CMU-ML-14-102

Carnegie Mellon

Towards Scalable Analysis of Images and Videos

Bin Zhao

SEPTEMBER 2014
CMU-ML-14-102

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Eric Xing, Chair
Tom Mitchell
Alex Hauptmann
Kristen Grauman

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

This research was sponsored by the National Science Foundation under grant numbers DBI0640543 and IIS1115313, the National Institutes of Health under grant number R01GM093156, the Air Force Office of Scientific Research under grant number FA95501010247, and the Office of Naval Research under grant number N000140910758.

Keywords: Image Classification, Video Summarization, Unusual Event Detection, Sparse Output Coding, Dynamic Sparse Coding, Online Dictionary Learning

To the memory of my grandfather (1927-2012)

Abstract

With widespread availability of low-cost devices capable of photo shooting and high-volume video recording, we are facing explosion of both image and video data. The sheer volume of such visual data poses both challenges and opportunities in machine learning and computer vision research.

In image classification, most of previous research has focused on small to medium-scale data sets, containing objects from dozens of categories. However, we could easily access images spreading thousands of categories. Unfortunately, despite the well-known advantages and recent advancements of multi-class classification techniques in machine learning, complexity concerns have driven most research on such super large-scale data set back to simple methods such as *nearest neighbor search*, *one-vs-one* or *one-vs-rest* approach. However, facing image classification problem with such huge task space, it is no surprise that these classical algorithms, often favored for their simplicity, will be brought to their knees not only because of the training time and storage cost they incur, but also because of the conceptual awkwardness of such algorithms in massive multi-class paradigms. Therefore, it is our goal to directly address the bigness of image data, not only the large number of training images and high-dimensional image features, but also the large task space. Specifically, we present algorithms capable of efficiently and effectively training classifiers that could differentiate tens of thousands of image classes.

Similar to images, one of the major difficulties in video analysis is also the huge amount of data, in the sense that videos could be hours long or even endless. However, it is often true that only a small portion of video contains important information. Consequently, algorithms that could automatically detect unusual events within streaming or archival video would significantly improve the efficiency of video analysis and save valuable human attention for only the most salient contents. Moreover, given lengthy recorded videos, such as those captured by digital cameras on mobile phones, or surveillance cameras, most users do not have the time or energy to edit the video such that only the most salient and interesting part of the original video is kept. To this end, we also develop algorithm for automatic video summarization, without human intervention. Finally, we further extend our research on video summarization into a supervised formulation, where users are asked to generate summaries for a subset of a class of videos of similar nature. Given such manually generated summaries, our algorithm learns the preferred storyline within the given class of videos, and automatically generates summaries for the rest of videos in the class, capturing the similar storyline as in those manually summarized videos.

Acknowledgments

I would like to thank my advisor Eric Xing, for his guidance in picking research problems, the tremendous help in designing milestones for the projects, his constant encouragement and numerous helpful advices. His exceptional vision in both research and real world application helped me greatly in shaping up this thesis. Eric has provided every help I would imagine from a research advisor, and I have truly enjoyed the experience of working with Eric.

I am indebted to Tom Mitchell, Alex Hauptmann, and Kristen Grauman for serving on my thesis committee and giving me helpful suggestions. I benefitted greatly from Tom's book and other publications, and his insightful suggestions and comments. Alex has also served on my Data Analysis Project committee, and I am truly grateful for his help on my research for both DAP and this thesis. Kristen's early work on unusual event detection is the motivation for analyzing video data in this thesis, and I am indebted to that. I am also grateful to Li Fei-Fei, for her truly helpful advices and suggestions in several projects we collaborated.

I learned a lot from former and current members of the SAILING Lab, and I owe much for the wonderful research comments they provided. I want to specially thank Gunhee Kim, Bin Shu, and Pengtao Xie for their expertise and help on projects.

I am grateful to many of CMU faculties, for their wonderful courses and talks during my five years in CMU. Also, I would like to thank Diane Stidle, Michelle Martin, Mallory Deptola, Sharon Cavlovich and the entire MLD staff for their help over the years.

I owe much to my former advisor in Tsinghua University, Changshui Zhang, for introducing me into the fascinating world of machine learning.

I want to thank my great friend and mentor, Victor Long, for all the inspiring discussions we have had, the time we spent together facing challenges, and for showing me the potential of machine learning in a brand new field.

Lastly, and most importantly, I am extremely grateful to my family. I would not be able to achieve anything without the constant encouragement and endless love from mom and dad. For my wife, Ting, I find it extremely difficult to use words to express my gratitude and appreciation for all the support and sacrifice during the years. Thanks for making my life colorful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The widespread availability of image / video capturing devices results in huge amount of visual data. There are almost 8 billion images on Flickr, and more than 16 billion photos on Instagram, a service started less than four years ago. Every minute, about 100 hours long of video is uploaded to YouTube, let alone the millions of surveillance cameras around the world recording videos around the clock. Clearly, the sheer size of visual data is way beyond human processing capability, and automatic analysis and understanding of these visual data is in great need.

1.1 Background and Motivation

With the ubiquitous installment of cameras on consumer electronics, such as smart phone, tablet, various wearable gadgets such as smart watch and glasses, and the global deployment of surveillance systems in space, air, sea, ground, and social media, the amount of unprocessed media data in the form of text, images and videos is massive. The sheer size of the media data is way beyond human processing capability, therefore computational means for automatic analysis, understanding, organization, and summarization of these data is greatly needed. In recent years, machine learning technologies such as topic models, latent space analysis [139], (e.g., Principal component analysis (PCA) [125], latent semantic indexing (LSI) [32], latent Dirichlet allocation (LDA) [16], Sparse Coding [77]), have led to a number of breakthroughs in automatic processing of large volume of textual information, to the extent that billions of text documents can be processed to extract trending topics and story lines [2]. However, such success is not matched in general visual data, such as images and videos. It is fair to say that much of the content in our multimedia universe remains "dark matter" to us.

Although machine learning technologies have been successfully used to tackle quite a few computer vision problems, the majority of these applications are in a much smaller scale than the real world demand.

Large-Scale Image Data. Image categorization / object recognition has been one of the most important research problems in the computer vision community. While most previous research on image categorization has focused on medium-scale data sets – for example, a limited number of class labels as seen in many popular benchmark data sets, there is recently a growing consensus that it is necessary to build general purpose object recognizers that are able to recognize

many more different classes of objects. In a modern era when prevalence of social media data and consumer-driven problems are inspiring attentions on data sets and tasks mimicking human intelligence in real world, a new dimension of bigness of machine learning and computer vision – big task space, merits serious attention due to a lack of scalable and robust new learning framework to meet the present and future challenges that are crumbing the decade-old classical approaches still in service, such as kNN or one-vs-all style classification [91]. Indeed, problems involving a massive amount of possible category labels (i.e., classes), in the order of tens or even hundreds of thousands, in addition to the large volume of data points and features, are easily within our reach. For example, ImageNet [35] for object recognition spans a total of 21841 classes. Similarly, TinyImage [127] contains 80 million 32×32 low resolution images, with each image loosely labeled with one of 75062 English nouns. Clearly, these are no longer artificial visual categorization problems created for machine learning, but instead more like a human-level cognition problem for real world object recognition with a much bigger set of objects. A natural way to formulate this problem is a *multi-class* or *multi-task* classification, but the seemingly standard formulation on such gigantic data set poses a completely new challenge both to computer vision and machine learning. Unfortunately, despite the well-known advantages and recent advancements of multi-class classification techniques [6, 13, 61] in machine learning, complexity concerns have driven most research on such super large-scale data set back to simple methods such as *nearest neighbor search* [18], *least square regression* [48], *one-vs-one* or *one-vs-rest* approach that requires learning tens of thousands of binary classifiers [82]. With such large number of classes, it is no surprise that classical algorithms such as one-vs-rest, one-vs-one, or kNN, often favored for their simplicity [18, 112], will be brought to their knees not only because of the training time and storage cost they incur [68], but also because of the conceptual awkwardness of such algorithms in massive multi-class paradigms. For example, facing 21841 classes in the ImageNet problem, should we go ahead and build 21841 classifiers each trained for 1-vs-21840 classification? Just imagine the resultant data imbalance issue at its extreme, let alone the terrible irregularities of the decision boundaries of such classifiers. Worse still, the number of classes can even grow further in the future.

Large-Scale Video Data. With the widespread availability, to both consumers and organizations, of low-cost devices capable of high-volume video recording, such as digital cameras on mobile phones, tablets, and soon, wearable gadgets such as glasses and watches; and various surveillance cameras and monitoring devices all over the world and in space, we are inundated with billion hours of video footage every day potentially containing events, people, and objects of context-dependent and time-space-sensitive interests. However, even to the creators / owners of such data, let alone all the people who are granted access for various purposes, the contents in all these videos remain *dark matter* in the data universe, because watching these recorded footage in real-time, or even playing at 2x or 4x speed is hardly possible and enjoyable. It is no surprise that with this increasing body of video data, which are largely left unedited and unstructured, all information therein are like trees falling in the forest — they are nearly impossible to access unless already been seen and indexed, an undertaking too tedious and time consuming for human, but an ideal challenge for machine intelligence.

In this thesis, we focus on developing machine learning algorithms to directly address the large scale of both image and video data. Specifically, we discuss how to efficiently and effectively train classifiers that could differentiate tens of thousands of image classes. At the same

Table 1.1: Thesis outline.

Part I – Large-Scale Image Classification
1. Large Scale Category Structure Aware Image Classification [149] (Chapter 3))
2. Sparse Output Coding for Scalable Visual Recognition [144] (Chapter 4)
Part II – Large Scale Video Understanding
1. Online Detection of Unusual Events in Videos via Dynamic Sparse Coding [148] (Chapter 5)
2. Quasi Real-Time Summarization for Consumer Videos [145] (Chapter 6)
3. Supervised Video Summarization: A Max Margin Approach (Chapter 7)

time, we explore how to detect interesting events from hour-long, or even endless video streams, such as surveillance videos, and automatically generate summaries for such large videos.

1.2 Thesis Statement

The thesis statement could be summarized into a single sentence as follows:

> **We aim to design machine learning algorithms to automatically analyze and understand large-scale image and video data.**

Specifically, we design algorithms to address the *bigness* in image categorization – not only in the form of large number of data points and/or high-dimensional features, but also the large task space. Our proposed algorithm scales to image collections with tens of thousands of classes. On the other hand, we also propose algorithms to address the *bigness* of video stream, with hours in temporal length, or even endless, and automatically distill such videos to identify interesting events and summarize its contents.

Consequently, this dissertation consists of two parts: (i) large scale image classification (Part I), (ii) large scale video understanding, with emphasis on unusual event detection, unsupervised video highlighting and supervised video summarization (Part II). Specifically, many vision tasks require a multi-class classifier to discriminate multiple categories, on the order of hundreds or thousands. Hence, Part I proposes algorithms for large scale image classification, that efficiently learns classifiers for image collections containing tens of thousands of classes. Part II focuses on understanding temporally long or even endless videos, by identifying interesting or unusual events automatically from any video stream without the requirement of human definition on normality or abnormality, automatically generating short trailer videos for unstructured and unedited videos, and understanding user preference through annotation on videos of similar nature for personalized and supervised video summarization.

Table 1.1 summarizes the outline of this thesis. In order to achieve the proposed thesis statement, we have completed the following projects.

Large Scale Image Classification (Part I). The main objective of the algorithms proposed in this part is to address the bigness in categorizing large collection of images, by using information from not only the images themselves, but also the hierarchical structure among image classes, usually available for large scale image collection.

- (Chapter 3) We propose a Map-Reduce based algorithm to effectively exploit hierarchical structure among image classes, for efficient and accurate image classification with large number of training images, features and classes.
- (Chapter 4)We present a fast and accurate algorithm for image categorization, scalable to problems with tens of thousands of classes, by turning high-cardinality multi-class categorization into a bit-by-bit decoding problem.

Large Scale Video Understanding (Part II). The goal of the algorithms proposed in this part is to automatically identify the interesting, unusual, salient portions of temporally long or even endless videos, such that the computationally expensive downstream processing such as activity recognition, object recognition, various detection tasks could focus on only small fraction of the original video, and achieve fast understanding of long videos.

- (Chapter 5) We develop an online algorithm to automatically detect unusual events within streaming or archival video, capable of handling long or even endless video streams, and capturing potential concept drift in videos.
- (Chapter 6) We extend the above unusual event detection algorithm, and develop a summarization algorithm that automatically compiles the most salient and informative portion of the video data for users, by automatically scanning through video stream, in an online fashion, to remove repetitive and uninteresting contents.
- (Chapter 7) We propose a max margin learning framework for supervised video summarization, where users are asked to generate summaries for a subset of a class of videos of similar nature. Given such manually generated summaries, our algorithm learns the preferred storyline within the given class of videos, and automatically generate summaries for the rest of videos in the class, capturing the similar storyline in those manually summarized videos.

Most chapters of this thesis have been published in [144, 145, 146, 147, 148, 149].

Chapter 2

Survey of Related Work

In this chapter, we review previous works that are related to this thesis. Specifically, we focus on the following two main directions: large-scale image classification, event detection and summarization for video streams.

2.1 Large-Scale Image Classification

Hierarchical classification: Facing large task space, one possible alternative that is attempted but still not popular is hierarchical classification (HC) [9, 11, 12, 22, 33, 38, 67, 151], which in principle can reduce the number of classification decisions to $O(\log K)$, where K is the number of leaf classes. However, HC faces remarkable difficulty in practice for large scale problems because of a number of undesirable intrinsic properties, such as sensitivity to reliability of near-root classifiers, error propagation along the tree path, over-heterogeneity of training data for near-root super classes, etc. Weighing the above issues with hierarchical classification, [50] introduced relaxed tree hierarchy, where a class can appear on both left child and right child of a node, with the ability to at least partially avoid error propagation. However, allowing a class to appear on both child nodes increases the computational complexity of the tree classifier. Moreover, [50] learns the relaxed tree structure and classifiers in a unified optimization framework, via alternating optimization. The fact that [50] needs to train classifiers multiple times in alternating optimization, renders it rather expensive computationally, especially for large-scale classification with big task space. Clearly, massive multi-class classification with the number of classes approaching or even surpassing human cognitive capability is an important yet under-addressed research problem, and requires new, out-of-box rethinking of classical approaches and more effective yet simple alternatives (we emphasize simplicity as for massive multi-class problems, any computationally intense methods would immediately fall out of favor by practitioners).

Label embedding: Another line of research related to this thesis is *label embedding* [59, 134, 136, 143], where each class is represented by a prototype vector in some subspace, into which all training data points are also projected. The projection is optimized such that data points are mapped close to their corresponding class prototype. Classification is then carried out using nearest neighbor search in the subspace. Similar to the idea of *label embedding*, *semantic output codes* [97] learns two projections which utilize a knowledge base of semantic properties

of label set to extrapolate to novel classes, where the first projection maps raw-input space into a semantic space, and then the second projection maps this semantic encoding to a class label.

Class structure aware multi-class classification: Various attempts in sharing information across related image categories have been explored. Early approaches stem from the neural networks, where the hidden layers are shared across different classes [23, 76]. Recent approaches transfer information across classes by regularizing the parameters of the classifiers across classes [7, 47, 90, 96, 108, 121, 123, 126]. Common to all these approaches is that experiments are always performed with relatively few classes [48]. It is unclear how these approaches would perform on super large-scale data sets containing thousands of image categories. Some of these approaches would encounter severe computational bottleneck when scaling up to thousands of classes [48].

Attributes: This line of research [10, 37, 46, 69, 70, 81, 101, 110, 128, 131] employs attribute descriptors, mid-level semantic visual concepts such as "short", "furry", "leg", etc., which are shareable across categories, to encode categorical information as image features. Each attribute could be response map of binary classifiers, and the object recognition task is carried out by utilizing multiple attributes as image features for training classifier. Specifically, the Meta-Class algorithm [10] employs label tree learning [9] to learn meta-classes, which are set of classes that can be easily separated from others. Meta-Class algorithm could be seen as generalization of one-vs-rest, where instead of using only one class as positive data, it selects a set of classes called Meta-Class as positive data in learning binary classifiers.

Image feature coding: Very recently, we have seen attempts in large-scale visual recognition through feature coding [75, 82, 103]. These approaches [75, 82, 103] focus on designing high-dimensional feature representation for images, where the classifier is trained using conventional one-vs-rest approach. Our proposed work focuses on learning multi-class classifiers and serves as an important complement to this line of research, in the sense that we could very easily combine our classification method with feature representations learned in [75, 82, 103] to yield even better results.

2.2 Event Detection and Summarization of Videos

Unusual event detection: Several attempts have been proposed in the literature on unsupervised unusual event detection in videos [1, 17, 54, 80, 133, 150]. Specifically, [54, 60, 120] studies the problem using tracking trajectories. However, even with the recent advances in tracking techniques, reliably tracking an object in a crowded video is still a very challenging research problem. Clustering methods [55, 150] have also been applied to detect unusual events, where the detection is carried out by finding spatially isolated clusters. The fact that these methods only run in batch mode severely limits their applicability. [1] proposes a simple yet effective approach that measures typical flow directions and speeds on a grid in the video frame to detect unusual events. This algorithm is good for detecting simple events such as moving in the wrong direction. [17] proposes a database indexing algorithm, where the problem is formulated as composing the new observed video data using spatio-temporal patches extracted from previous visual examples. Regions in the query video that can be composed using large contiguous chunks of data from the example database are considered normal. Although this algorithm shows good performance in

discriminating complex motions, it faces scalability issues as its time and memory complexity is linear in the size of the example database. Finally, [65] utilizes a space-time Markov random field to detect unusual events, where an MRF model is built for usual events and those events that could not be described with the learned model is considered as unusual.

Video Summarization: Previous research on video summarization and abstraction has mainly focused on edited videos, e.g., movies, news, and sports, which are highly structured [94, 129]. For example, a movie could be naturally divided into scenes, each formed by one or more shots taken place at the same site, and each shot is further composed of frames with smooth and continuous motions. However, consumer generated videos and surveillance videos lack such structure, often rendering previous research not directly applicable.

Key frame based methods compose video summary as a collection of salient images (key frames) picked from the original video. Various strategies haven been studied, including shot boundary detection [41], color histogram [141], motion stability [137, 141], clustering [56], curve splitting [34], and frame self-expressiveness [44]. However, isolated and uncorrelated still images, without smooth temporal continuation, are not best suited to help the viewer understand the original video. Moreover, [78] proposes a saliency based method, which trains a linear regression model to predict importance score for each frame in egocentric videos [78]. However, special features designed in [78] limit its applicability only to videos generated by wearable cameras.

Besides picking frames from the original video, methods creating new image not present in the original video have also been studied [4, 24, 52, 83, 106, 111, 117], where a panoramic image is generated from a few consecutive frames having some important content. However, the number of consecutive frames from original video used to construct such panoramic image is limited by occlusion between objects from different frames. Consequently, these approaches generally assume short clips with few objects.

Finally, summaries composed by a collection of video segments, have been studied for edited videos. Specifically, [104] used scene boundary detection, dialogue analysis, and color histogram to produce trailer for a feature film. [5] and [79] extracted important segments from sports and news programs utilizing special characteristics of these videos, including fixed scene structures, dominant locations, and backgrounds. Moreover, [122] and [118] utilized closed caption and speech recognition to transform video summarization into a text summarization problem and generated summaries using natural language processing techniques. However, the large body of consumer generated videos and surveillance videos usually have no such special structure, nor audio information at all.

Event Recognition: Another line of research related to this thesis is activity / event recognition [53, 58, 71, 87, 95, 100, 114], such as recognizing the event of bride walking down the aisle in wedding videos. However, in activity recognition, recognizing each unique type of activity is treated as a separate problem, in the sense that each activity type requires a separate set of training data, and the learning procedure aims to obtain classifier that recognizes only the specific activity. On the other hand, in our supervised video summarization formulation, the manually generated summary video does not differentiate different types of events. In particular, the provided summary video only indicates which portions of the original video should be incorporated into the summary, without specific information on the event type for each segment included in the summary. Moreover, our supervised video summarization formulation does not

require knowledge on the number of unique events, which is always assumed available in activity recognition literature.

Part I

Large Scale Image Classification

Part I Large Scale Image Classification

In this part, we discuss classification methods for image collections with large numbers of classes, huge amounts of training images, and high dimensional feature representation for each image. Specifically, we focus on both accuracy and scalability of learning algorithms for such large image collections.

This part consists of two chapters. First, we observe that one key characteristic for large scale image classification is the availability of hierarchical structure among different image classes, resembling how human cognitive system stores visual knowledge. Specifically, image classes in such problems are rarely organized in a flat fashion, but instead with a taxonomical hierarchical structure, such as a tree or DAG. This hierarchical structure could either stem from available lexical database of English nouns, such as WordNet, or obtained through structure learning algorithms [9, 38, 50, 113, 151]. This chapter studies the problem of how to effectively utilize such hierarchical structure among image classes, for efficient and accurate image classification with large number of training images, features and classes.

Second, we investigate even larger scale image classification problems, where the number of image classes in the collection could reach the scale of tens of thousands or even more. Our proposed approach is based on the error correcting output coding framework, where the optimal coding matrix is learned through a sparsity constrained optimization problem, followed by carefully designed probabilistic decoding to assign labels to testing images. The algorithm discussed in this chapter is a principled way for large-scale multi-class classification, by turning high-cardinality multi-class categorization into a bit-by-bit decoding problem. We have shown in experiments that this method scales up to image collections with tens of thousands of classes.

Chapter 3

Large Scale Category Structure Aware Image Classification

One key characteristic for image classification with a large number of categories is the hierarchical structure among different image classes, resembling how human cognitive system stores visual knowledge. Specifically, image classes in such problems are rarely organized in a flat fashion, but instead with a taxonomical hierarchical structure, such as a tree or DAG. This hierarchical structure could either stem from available lexical database of English nouns, such as WordNet, or obtained through structure learning algorithms [9, 38, 50, 113, 151]. In this chapter, we study the problem of how to effectively utilize such hierarchical structure among image classes, for efficient and accurate image classification with large number of training images, features and classes.

3.1 Introduction

Figure 3.1(a) shows an example of hierarchical structure among image classes, organized as a tree, where leaf nodes are individual categories, and each internal node denotes the cluster of categories corresponding to the leaf nodes in the subtree rooted at the given node. As human cognition of complex visual world benefits from underlying semantic relationships between object classes, we believe a machine learning system can and should leverage such information as well for better performance. Specifically, we argue that instead of formulating the recognition task as a flat classification problem, where each category is treated equally and independently, a better strategy is to utilize the rich information residing in the concept hierarchy among image categories to train a system that couples all different recognition tasks over different categories.

To the best of our knowledge, our attempt in this chapter represents an initial foray to systematically utilizing information residing in concept hierarchy, for multi-way classification on super large-scale image data sets. More precisely, our approach utilizes the concept hierarchy in two aspects: loss function and feature selection. First, the loss function used in our formulation weighs differentially for different misclassification outcomes: misclassifying an image to a category that is close to its true identity should receive less penalty than misclassifying it to a totally unrelated one. Second, in an image classification problem with thousands of categories, it is not realistic

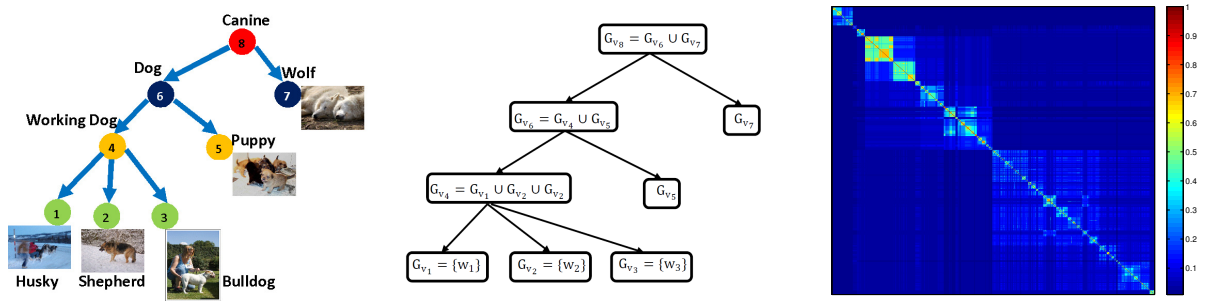

Figure 3.1: (a) Image category hierarchy in ImageNet; (b) Overlapping group structure; (c) Semantic relatedness measure between image categories.

to assume that all of the classes share the same set of relevant features. That is to say, a subset of highly related categories may share a common set of relevant features, whereas weakly related categories are less likely to be affected by the same features. Consequently, the image categorization problem is formulated as *augmented logistic regression with overlapping-group-lasso regularization*. The corresponding optimization problem involves a non-smooth convex objective function represented as summation over all training examples. To solve this optimization problem, we introduce the *Accelerated Parallel ProximaL gradiEnT (APPLET)* method, which tackles the non-smoothness of overlapping-group-lasso penalty via proximal gradient [27, 62], and the huge number of training examples by Map-Reduce parallel computing [29]. Therefore, the contributions made in this chapter are: (1) We incorporate the semantic relationships between object classes, into an augmented multi-class logistic regression formulation, regularized by the overlapping-group-lasso penalty. The sheer size of the image data set that our formulation is designed to tackle singles out our work from previous attempts on multi-class classification, or transfer learning. (2) We propose a proximal gradient based method for solving the resulting non-smooth optimization problem, where the super large scale of the problem is tackled by map-reduce parallel computation.

3.2 Problem Formulation

In this section, we describe the detailed formulation of our proposed category structure aware image classification, where hierarchical structure among image classes is effectively utilized in loss function definition and feature selection.

3.2.1 Hierarchical Structure among Image Classes

We will use the hierarchical structure in ImageNet, stemmed from WordNet, as an example. However, it should be noted that our proposed approach could be applied to any tree structure among image classes. Specifically, image categories in ImageNet are interlinked by several types of relations, with the "IS-A" relation being the most comprehensive and useful [36], resulting in

a tree hierarchy over image categories. For example, the 'husky' category follows a path in the tree composed of 'working dog', 'dog', 'canine', etc. In this work, we use the distance between two nodes in the tree to define the difference between the two corresponding image categories. Consequently, in the category hierarchy in ImageNet, each internal node near the bottom of the tree shows that the image categories of its subtree are highly correlated, whereas the internal node near the root represents relatively weaker correlations among the categories in its subtree.

The class hierarchy provides a measure of relatedness between image classes. Misclassifying an image to a category that is close to its true identity should receive less penalty than misclassifying it to a totally unrelated one. For example, although horses are not exactly ponies, we expect the loss for classifying a "pony" as a "horse" to be lower than classifying it as a "car". Instead of using 0-1 loss as in conventional image categorization, which treats image categories equally and independently, our approach utilizes a loss function that is aware of the category hierarchy.

Moreover, highly related image categories are more likely to share common visual patterns. For example, in Figure 3.1(a), *husky* and *shepherd* share similar object shape and texture. Consequently, recognition of these related categories are more likely to be affected by the same features. In this work, we regularize the sparsity pattern of weight vectors for related categories. This is equivalent to learning a low dimensional representation that is shared across multiple related categories.

3.2.2 Logistic Regression with Category Structure

Given N training images, each represented as a J-dimensional input vector and belonging to one of the K categories. Let $\mathbf{X}$ denote the $J \times N$ input matrix, where each column corresponds to an instance. Similarly, let $\mathbf{Y}$ denote the $N \times 1$ output vector, where each element corresponds to the label for an image. Multi-class logistic regression defines a weight vector $\mathbf{w}_k$ for each class $k \in \{1, \ldots, K\}$ and classifies example $\mathbf{x}$ by $y^* = \arg\max_{y \in \{1,\ldots,k\}} P(y|\mathbf{x}, \mathbf{W})$, with the conditional likelihood computed as

$$P(y_i|\mathbf{x}_i, \mathbf{W}) = \frac{\exp(\mathbf{w}_{y_i}^T \mathbf{x}_i)}{\sum_k \exp(\mathbf{w}_k^T \mathbf{x}_i)} \tag{3.1}$$

The optimal weight vectors $\mathbf{W}^* = [\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*]$ are

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \left(-\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i, \mathbf{W}) + \lambda \Omega(\mathbf{W}) \right) \tag{3.2}$$

where $\Omega(\mathbf{W})$ is a regularization term defined on $\mathbf{W}$ and λ is the regularization parameter.

Augmented Soft-Max Loss Function

Using the tree hierarchy on image categories, we could calculate a semantic relatedness (a.k.a. similarity) matrix $\mathbf{S} \in \mathbb{R}^{K \times K}$ over all categories, where $\mathbf{S}_{ij}$ measures the semantic relatedness of class i and j. Using the semantic relatedness measure, the likelihood of $\mathbf{x}_i$ belonging to category

y_i could be modified as follows

$$\hat{P}(y_i|\mathbf{x}_i, \mathbf{W}) \propto \sum_{r=1}^{K} \mathbf{S}_{y_i,r} P(r|\mathbf{x}_i, \mathbf{W}) \propto \sum_{r=1}^{K} \mathbf{S}_{y_i,r} \frac{\exp(\mathbf{w}_r^T \mathbf{x}_i)}{\sum_k \exp(\mathbf{w}_k^T \mathbf{x}_i)} \propto \sum_{r=1}^{K} \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \quad (3.3)$$

Since $\sum_{r=1}^{K} \hat{P}(r|\mathbf{x}_i, \mathbf{W}) = 1$, consequently,

$$\hat{P}(y_i|\mathbf{x}_i, \mathbf{W}) = \frac{\sum_{r=1}^{K} \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i)}{\sum_{r=1}^{K} \sum_{k=1}^{K} \mathbf{S}_{k,r} \exp(\mathbf{w}_r^T \mathbf{x}_i)} \quad (3.4)$$

For the special case where the semantic relatedness matrix $\mathbf{S}$ is an identity matrix, meaning each class is only related to itself, Eq. (3.4) simplifies to Eq. (3.1). Using this modified softmax loss function, the image categorization problem could be formulated as

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \left[\log \left(\sum_r \sum_k \mathbf{S}_{k,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) - \log \left(\sum_r \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) \right] + \lambda \Omega(\mathbf{W}) \quad (3.5)$$

Semantic Relatedness Matrix

To compute the semantic relatedness matrix $\mathbf{S}$ in the above formulation, we first define a metric measuring the semantic distance between image categories. A simple way to compute semantic distance in a structure such as the one provided by ImageNet is to utilize the paths connecting the two corresponding nodes to the root node. Following [21] we define the semantic affinity Q_{ij} between class i and class j as the number of nodes shared by their two parent branches, divided by the length of the longest of the two branches

$$Q_{ij} = \text{intersect}(P_i, P_j) / \max(\text{length}(P_i), \text{length}(P_j)) \quad (3.6)$$

where P_i is the path from root node to node i and $\text{intersect}(P_i, P_j)$ counts nodes shared by two paths P_i and P_j. We then construct the semantic relatedness matrix

$$\mathbf{S} = \exp(-\kappa(\mathbf{E} - \mathbf{Q})) \quad (3.7)$$

where κ is a constant controlling the decay factor of semantic relatedness with respect to semantic distance, and $\mathbf{E} = \mathbf{e}_K \mathbf{e}_K^\top$ is the $K \times K$ all-one matrix. Figure 3.1(c) shows the semantic relatedness matrix computed with $\kappa = 5$.

3.2.3 Tree-Guided Sparse Feature Coding

In ImageNet, image categories are grouped at multiple granularity as a tree hierarchy. As illustrated in Section 3.2.1, the image categories in each internal node are likely to be influenced by a common set of features. In order to achieve this type of structured sparsity at multiple levels of the hierarchy, we utilize an overlapping-group-lasso penalty recently proposed in [66] for genetic association mapping problem, where the goal is to identify a small number of SNPs (inputs) out of millions of SNPs that influence phenotypes (outputs) such as gene expression measurements.

Specifically, given the tree hierarchy $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ over image categories, each node $v \in \mathcal{V}$ of tree $\mathcal{T}$ is associated with group G_v, composed of all leaf nodes in the subtree rooted at v, as illustrated in Figure 3.1(b). Clearly, each group G_v is a subset of the power set of $\{1, \ldots, K\}$. Given these groups $\mathcal{G} = \{G_v\}_{v \in \mathcal{V}}$ of categories, we define the following overlapping-group-lasso penalty [66]:

$$\Omega(\mathbf{W}) = \sum_j \sum_{v \in \mathcal{V}} \gamma_v ||\mathbf{w}_{jG_v}||_2 \tag{3.8}$$

where $\mathbf{w}_{jG_v}$ is the vector of weight coefficients $\{w_{jk}, k \in G_v\}$ for input $j \in \{1, \ldots, J\}$ associated with categories in G_v, and each group G_v is associated with weight γ_v that reflects the strength of correlation within the group. It should be noted that we do not require groups in $\mathcal{G}$ to be mutually exclusive, and consequently, each leaf node would belong to multiple groups at various granularity.

Inserting the above overlapping-group-lasso penalty into (3.5), we formulate the category structure aware image categorization as follows:

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \left[\log \left(\sum_r \sum_k \mathbf{S}_{k,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) - \log \left(\sum_r \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) \right] + \lambda \sum_j \sum_{v \in \mathcal{V}} \gamma_v ||\mathbf{w}_{G_v}^j||_2 \tag{3.9}$$

3.3 Methods

The challenge in solving problem (3.9) lies in two facts: the non-separability of $\mathbf{W}$ in the non-smooth overlapping-group-lasso penalty $\Omega(\mathbf{W})$, and the huge number of training examples. Conventionally, to handle the non-smoothness of $\Omega(\mathbf{W})$, we could reformulate the problem as either *second order cone programming (SOCP)* or *quadratic programming (QP)* [124]. However, the state-of-the-art approach for solving *SOCP* and *QP* based on *interior point method* requires solving a Newton system to find search direction, and is computationally very expensive even for moderate-sized problems. Moreover, due to the huge number of samples in the training set, off-the-shelf optimization solvers are too slow to be used.

In this work, we adopt a proximal-gradient method to handle the non-smoothness of $\Omega(\mathbf{W})$. Specifically, we first reformulate the overlapping-group-lasso penalty $\Omega(\mathbf{W})$ into a $\max$ problem over auxiliary variables using dual norm, and then introduce its smooth lower bound [27, 62]. Instead of optimizing the original non-smooth penalty, we run the *accelerated gradient descent* method [92] under a Map-Reduce framework [29] to optimize the smooth lower bound. The proposed approach enjoys a fast convergence rate and low per-iteration complexity.

3.3.1 Reformulate the Penalty

For referring convenience, we number the elements in the set $\mathcal{G} = \{G_v\}_{v \in \mathcal{V}}$ as $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_{|\mathcal{G}|}\}$ according to an arbitrary order, where $|\mathcal{G}|$ denotes the total number of elements in $\mathcal{G}$. For each input j and group $\mathbf{g}_i$ associated with $\mathbf{w}_{j\mathbf{g}_i}$, we introduce a vector of auxiliary variables $\boldsymbol{\alpha}_{j\mathbf{g}_i} \in \mathbb{R}^{|\mathbf{g}_i|}$. Since the dual norm of L_2 norm is also an L_2 norm, we can reformulate $||\mathbf{w}_{j\mathbf{g}_i}||_2$ as

$$||\mathbf{w}_{j\mathbf{g}_i}||_2 = \max_{||\boldsymbol{\alpha}_{j\mathbf{g}_i}||_2 \leq 1} \boldsymbol{\alpha}_{j\mathbf{g}_i}^T \mathbf{w}_{j\mathbf{g}_i} \tag{3.10}$$

Moreover, define the following $\sum_{\mathbf{g}\in\mathcal{G}} |\mathbf{g}| \times J$ matrix

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\alpha}_{1\mathbf{g}_1} & \dots & \boldsymbol{\alpha}_{J\mathbf{g}_1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}_{1\mathbf{g}_{|\mathcal{G}|}} & \dots & \boldsymbol{\alpha}_{J\mathbf{g}_{|\mathcal{G}|}} \end{pmatrix} \tag{3.11}$$

in domain

$$\mathcal{O} = \{\mathbf{A}| \, ||\boldsymbol{\alpha}_{j\mathbf{g}_i}||_2 \leq 1, \forall j \in \{1, \dots, J\}, \mathbf{g}_i \in \mathcal{G}\} \tag{3.12}$$

Following [27], the overlapping-group-lasso penalty in (3.9) can be equivalently reformulated as

$$\Omega(\mathbf{W}) = \sum_j \sum_i \gamma_i \max_{||\boldsymbol{\alpha}_{j\mathbf{g}_i}||_2 \leq 1} \boldsymbol{\alpha}_{j\mathbf{g}_i}^T \mathbf{w}_{j\mathbf{g}_i} = \max_{A\in\mathcal{O}} \langle \mathbf{C}\mathbf{W}^T, \mathbf{A}\rangle \tag{3.13}$$

where $i = 1, \dots, |\mathcal{G}|$, $j = 1, \dots, J$, $\mathbf{C} \in \mathbb{R}^{\sum_{\mathbf{g}\in\mathcal{G}} |\mathbf{g}| \times K}$, and $\langle \mathbf{U}, \mathbf{V}\rangle = \mathrm{Tr}(\mathbf{U}^T\mathbf{V})$ is the inner product of two matrices. Moreover, the matrix $\mathbf{C}$ is defined with rows indexed by $(s, \mathbf{g}_i)$ such that $s \in \mathbf{g}_i$ and $i \in \{1, \dots, |\mathcal{G}|\}$, columns indexed by $k \in \{1, \dots, K\}$, and the value of the element at row $(s, \mathbf{g}_i)$ and column k set to $\mathbf{C}_{(s,\mathbf{g}_i),k} = \gamma_i$ if $s = k$ and 0 otherwise.

After the above reformulation, (3.13) is still a non-smooth function of $\mathbf{W}$, and this makes the optimization challenging. To tackle this problem, we introduce an auxiliary function [27, 62] to construct a smooth approximation of (3.13). Specifically, our smooth approximation function is defined as:

$$f_\mu(\mathbf{W}) = \max_{\mathbf{A}\in\mathcal{O}} \langle \mathbf{C}\mathbf{W}^T, \mathbf{A}\rangle - \mu d(\mathbf{A}) \tag{3.14}$$

where μ is the positive smoothness parameter and $d(\mathbf{A})$ is an arbitrary smooth strongly-convex function defined on $\mathcal{O}$. The original penalty term can be viewed as $f_\mu(\mathbf{W})$ with $\mu = 0$. Since our algorithm will utilize the optimal solution $\mathbf{W}^*$ to (3.14), we choose $d(\mathbf{A}) = \frac{1}{2}||\mathbf{A}||_F^2$ so that we can obtain the closed form solution for $\mathbf{A}^*$. Clearly, $f_\mu(\mathbf{W})$ is a lower bound of $f_0(\mathbf{W})$, with the gap computed as

$$D = \max_{\mathbf{A}\in\mathcal{O}} d(\mathbf{A}) = \max_{\mathbf{A}\in\mathcal{O}} \frac{1}{2}||\mathbf{A}||_F^2 = \frac{1}{2} J|\mathcal{G}| \tag{3.15}$$

Theorem 1 *For any $\mu > 0$, $f_\mu(\mathbf{W})$ is a convex and continuously differentiable function in $\mathbf{W}$, and the gradient of $f_\mu(\mathbf{W})$ can be computed as $\nabla f_\mu(\mathbf{W}) = \mathbf{A}^{*T}\mathbf{C}$, where A^* is the optimal solution to (3.14).*

According to Theorem 1, $f_\mu(\mathbf{W})$ is a smooth function for any $\mu > 0$, with a simple form of gradient and can be viewed as a smooth approximation of $f_0(\mathbf{W})$ with the maximum gap of μD. Finally, the optimal solution $\mathbf{A}^*$ of (3.14) is composed of $\boldsymbol{\alpha}^*_{j\mathbf{g}_i} = S(\frac{\gamma_i \mathbf{w}_{j\mathbf{g}_i}}{\mu})$, where S is the shrinkage operator defined as follows:

$$S(\mathbf{u}) = \begin{cases} \frac{\mathbf{u}}{||\mathbf{u}||_2}, & ||\mathbf{u}||_2 > 1 \\ \mathbf{u}, & ||\mathbf{u}||_2 \leq 1 \end{cases} \tag{3.16}$$

3.3.2 Accelerated Parallel Gradient Method

Given the smooth approximation of $\Omega(\mathbf{W})$ in (3.14) and the corresponding gradient presented in Theorem 1, we could apply *gradient descent* method to solve the problem. Specifically, we replace the overlapping-group-lasso penalty in (3.9) with its smooth approximation $f_\mu(\mathbf{W})$ to obtain the following optimization problem

$$\min_{\mathbf{W}} \tilde{f}(\mathbf{W}) = g(\mathbf{W}) + \lambda f_\mu(\mathbf{W}) \tag{3.17}$$

where

$$g(\mathbf{W}) = \sum_{i=1}^{N} \left[\log \left(\sum_r \sum_k \mathbf{S}_{k,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) - \log \left(\sum_r \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i) \right) \right] \tag{3.18}$$

is the augmented logistic regression loss function. The gradient of $g(\mathbf{W})$ w.r.t. $\mathbf{w}_k$ could be calculated as follows

$$\frac{\partial g(\mathbf{W})}{\partial \mathbf{w}_k} = \sum_{i=1}^{N} \mathbf{x}_i \left[\frac{\sum_q \mathbf{S}_{k,q} \exp(\mathbf{w}_k^T \mathbf{x}_i)}{\sum_r \sum_q \mathbf{S}_{r,q} \exp(\mathbf{w}_r^T \mathbf{x}_i)} - \frac{\mathbf{S}_{y_i,k} \exp(\mathbf{w}_k^T \mathbf{x}_i)}{\sum_r \mathbf{S}_{y_i,r} \exp(\mathbf{w}_r^T \mathbf{x}_i)} \right] \tag{3.19}$$

Therefore, the gradient of $g(\mathbf{W})$ w.r.t. to $\mathbf{W}$ could be computed as

$$\nabla g(\mathbf{W}) = [\frac{\partial g(\mathbf{W})}{\partial \mathbf{w}_1}, \ldots, \frac{\partial g(\mathbf{W})}{\partial \mathbf{w}_K}] \tag{3.20}$$

According to Theorem 1, the gradient of $\tilde{f}(\mathbf{W})$ is given by

$$\nabla \tilde{f}(\mathbf{W}) = \nabla g(\mathbf{W}) + \lambda \mathbf{A}^{*T} \mathbf{C} \tag{3.21}$$

Although $\tilde{f}(\mathbf{W})$ is a smooth function of $\mathbf{W}$, it is represented as a summation over all training samples. Consequently, $\nabla \tilde{f}(\mathbf{W})$ could only be computed by summing over all N training samples. Due to the huge number of samples in the training set, we adopt a Map-Reduce parallel framework [29] to compute $\nabla g(\mathbf{W})$ as shown in Eq.(3.19). While standard gradient schemes have a slow convergence rate, they can often be accelerated. This stems from the pioneering work of Nesterov in [92], which is a deterministic algorithm for smooth optimization. In this paper, we adopt this accelerated gradient method, and the whole algorithm is shown in Algorithm 1.

3.4 Experiments

In this section, we test the performance of *APPLET* on a subset of ImageNet used in ILSVRC10, containing 1.2 million images from 1000 categories, divided into distinct portions for training, validation and test. The number of images for each category ranges from 668 to 3047. We use the provided validation set for parameter selection and the final results are obtained on the test set.

Algorithm 1 Accelerated Parallel ProximaL gradiEnT method (APPLET)

Input: $\mathbf{X}$, $\mathbf{Y}$,$\mathbf{C}$, desired accuracy ϵ, step parameters $\{\eta_t\}$
Initialization: $\mathbf{B}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$, until convergence **do**
  Map-step: Distribute data to M cores $\{\mathcal{X}_1, \ldots, \mathcal{X}_M\}$, compute in parallel $\nabla g_m(\mathbf{B}_{t-1})$ for $\mathcal{X}_m$
  Reduce-step:
  (1) $\nabla \tilde{f}(\mathbf{B}_{t-1}) = \sum_{m=1}^{M} \nabla g_m(\mathbf{B}_{t-1}) + \lambda \mathbf{A}^{*T}\mathbf{C}$
  (2) $\mathbf{W}_t = \mathbf{B}_{t-1} - \eta_t \nabla \tilde{f}(\mathbf{B}_{t-1})$
  (3) $\mathbf{B}_t = \mathbf{W}_t + \frac{t-1}{t+2}(\mathbf{W}_t - \mathbf{W}_{t-1})$
end for
Output: $\hat{\mathbf{W}} = \mathbf{W}_t$

Before presenting the classification results, we'd like to make clear that the goal and contributions of this work is different from the aforementioned approaches proposed in ILSVRC10. Those approaches were designed to enter a performance competition, where heavy feature engineering and post processing (such as ad hoc voting for multiple algorithms) were used to achieve high accuracy. Our work, on the other hand, looks at this problem from a different angle, focusing on principled methodology that explores the benefit of utilizing class structure in image categorization and proposing a model and related optimization technique to properly incorporate such information. We did not use the full scope of all the features, and post processing schemes to boost our classification results as the ILSVRC10 competition teams did. Therefore we argue that the results of our work is not directly comparable with the ILSVRC10 competitions.

3.4.1 Image Features

Each image is resized to have a max side length of 300 pixels. SIFT [84] descriptors are computed on 20×20 overlapping patches with a spacing of 10 pixels. Images are further downsized to $\frac{1}{2}$ of the side length and then $\frac{1}{4}$ of the side length, and more descriptors are computed. We then perform k-means clustering on a random subset of 10 million SIFT descriptors to form a visual vocabulary of 1000 visual words. Using this learned vocabulary, we employ *Locality-constrained Linear Coding (LLC)* [132], which has shown state-of-the-art performance on several benchmark data sets, to construct a vector representation for each image. Finally, a single feature vector is computed for each image using max pooling on a spatial pyramid [74]. The pooled features from various locations and scales are then concatenated to form a spatial pyramid representation of the image. Consequently, each image is represented as a vector in a 21,000 dimensional space.

3.4.2 Evaluation Criteria

We adopt the same performance measures used in ILSVRC10. Specifically, for every image, each tested algorithm will produce a list of 5 object categories in the descending order of confidence. Performance is measured using the top-n error rate, $n = 1, \ldots, 5$ in our case, and two error measures are reported. The first is a **flat error** which equals 1 if the true class is not within the

n most confident predictions, and 0 otherwise. The second is a **hierarchical error**, reporting the minimum height of the lowest common ancestors between true and predicted classes. For each of the above two criteria, the overall error score for an algorithm is the average error over all test images.

Table 3.1: Classification results (both flat and hierarchical errors) of various algorithms.

	Flat Error					Hierarchical Error				
Algorithm	Top 1	Top 2	Top 3	Top 4	Top 5	Top 1	Top 2	Top 3	Top 4	Top 5
LR	0.797	0.726	0.678	0.639	0.607	8.727	6.974	5.997	5.355	4.854
ALR	0.796	0.723	0.668	0.624	0.587	8.259	6.234	5.061	4.269	3.659
GroupLR	0.786	0.699	0.642	0.600	0.568	7.620	5.460	4.322	3.624	3.156
APPLET	0.779	0.698	0.634	0.589	0.565	7.208	4.985	3.798	3.166	3.012

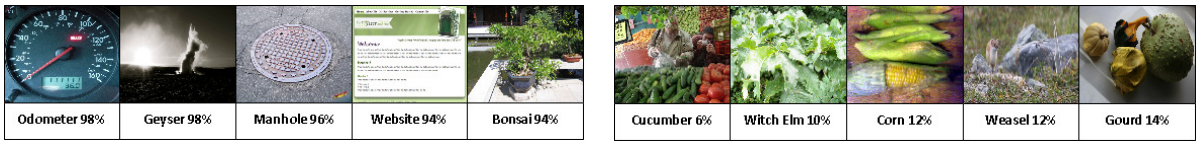

Figure 3.2: Left: image classes with highest accuracy. Right: image classes with lowest accuracy.

3.4.3 Comparisons & Classification Results

We have conducted comprehensive performance evaluations by testing our method under different circumstances. Specifically, to better understand the effect of augmenting logistic regression with semantic relatedness and use of overlapping-group-lasso penalty to enforce group level feature selection, we study the model adding only augmented logistic regression loss and adding only overlapping-group-lasso penalty separately, and compare with the *APPLET* method. We use the conventional L_2 regularized logistic regression [14] as baseline. The algorithms that we evaluated are listed below: (1)L_2 regularized logistic regression (LR) [14]; (2) Augmented logistic regression with L_2 regularization (ALR); (3) Logistic regression with overlapping-group-lasso regularization (GroupLR); (4) Augmented logistic regression with overlapping-group-lasso regularization (APPLET).

Table 3.1 presents the classification results of various algorithms. According to the classification results, we could clearly see the advantage of *APPLET* over conventional logistic regression, especially on the top-5 error rate. Specifically, comparing the top-5 error rate, *APPLET* outperforms *LR* by a margin of 0.04 on flat loss, and a margin of 1.84 on hierarchical loss. It should be noted that hierarchical error is measured by the height of the lowest common ancestor in the hierarchy, and moving up a level can more than double the number of descendants. Table 3.1 also compares the performance of *ALR* with *LR*. Specifically, *ALR* outperforms *LR* slightly when using the top-1 prediction results. However, on top-5 prediction results, *ALR* performs clearly better than *LR*. Similar phenomenon is observed when comparing the classification results of

Table 3.2: Example prediction results of *APPLET* and *LR*. Numbers indicate the **hierarchical error** of the misclassification, defined in Section 3.4.2.

True class	laptop	linden	gordon setter	gourd	bullfrog	volcano	odometer	earthworm
APPLET	laptop(0)	live oak(3)	Irish setter(2)	acorn(2)	woodfrog(2)	volcano(0)	odometer(0)	earthworm(0)
LR	laptop(0)	log wood(3)	alp(11)	olive(2)	water snake(9)	geyser(4)	odometer(0)	slug(8)

GroupLR with *LR*. Moreover, Figure 3.2 shows the image categories with highest and lowest classification accuracy.

One key reason for introducing the augmented loss function is to ensure that the predicted image class falls not too far from its true class on the semantic hierarchy. Results in Table 3.2 demonstrate that even though *APPLET* cannot guarantee to make the correct prediction on each image, it produces labels that are closer to the true one than *LR*, which generates labels far from correct ones.

As shown in Table 3.1, a systematic reduction in classification error using *APPLET* shows that acknowledging semantic relationships between image classes enables the system to discriminate at more informative semantic levels. Moreover, results in Table 3.2 demonstrate that classification results of *APPLET* can be significantly more informative, as labeling a "bullfrog" as "woodfrog" gives a more useful answer than "water snake", as it is still correct at the "frog" level.

3.4.4 Effects of λ and κ on the Performance of *APPLET*

We present in Figure 3.3 how categorization performance scales with λ and κ. According to Figure 3.3, *APPLET* achieves lowest categorization error around $\lambda = 0.01$. Moreover, the error rate increases when λ is larger than 0.1, when excessive regularization hampers the algorithm from differentiating semantically related categories. Similarly, *APPLET* achieves best performance with $\kappa = 5$. When κ is too small, a large number of categories are mixed together, resulting in a much higher flat loss. On the other hand, when $\kappa \geq 50$, the semantic relatedness matrix is close to diagonal, resulting in treating all categories independently, and categorization performance becomes similar as *LR*.

3.5 Summary

The *APPLET* algorithm represents the first step towards efficient large scale image classification, with large number of training images, hundreds of thousands of features for each image, as well as thousands of different image classes. Our proposed approach is based on the observation that image classes in such problems are rarely organized in a flat fashion, but instead with a taxonomical hierarchical structure. Our formulation effectively utilizes such hierarchical structure to

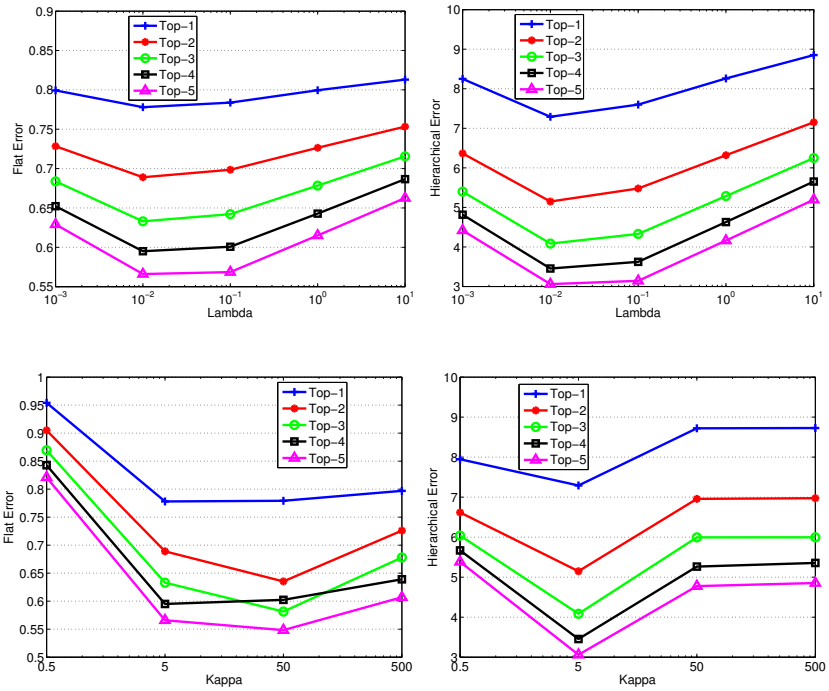

Figure 3.3: Classification results (flat error and hierarchical error) of *APPLET* with various λ and κ.

enforce the assumption that misclassifying an image to a category that is close to its true identity should receive less penalty than misclassifying it to a totally unrelated one, and a subset of highly related categories may share a common set of relevant features, whereas weakly related categories are less likely to be affected by the same features. The sheer size of the problem considered here singles out our work from any previous works on multi-way classification or transfer learning. Empirical study using 1.2 million training images from 1000 categories demonstrates the effectiveness and promise of our proposed approach. The next step is further scaling up to even larger concept space, with tens or even hundreds of thousands of different classes, which we will discuss in the next chapter.

Chapter 4

Sparse Output Coding for Scalable Visual Recognition

This chapter proposes a multi-class classification method that is both accurate and fast when facing a large number of categories, in the order of tens or even hundreds of thousands. Specifically, we propose *sparse output coding (SpOC)*, a principled way for large-scale multi-class classification, by turning high-cardinality multi-class categorization into a bit-by-bit decoding problem. Algorithmicaly, sparse output coding is composed of two steps: efficient coding matrix learning with scalability to tens of thousands of classes, and probabilistic decoding. Empirical results on object recognition and scene classification demonstrate the effectiveness of our proposed approach.

4.1 Introduction

For a K class problem, *error correcting output coding (ECOC)* [3] consists of two stages: coding and decoding. An output code $\mathbf{B}$ is a matrix of size $K \times L$ over $\{-1, 0, +1\}$ where each row of $\mathbf{B}$ corresponds to a class $y \in \mathcal{Y} = \{1, \ldots, K\}$. Each column $\boldsymbol{\beta}_l$ of $\mathbf{B}$ defines a partition of $\mathcal{Y}$ into three disjoint sets: positive partition ($+1$ in $\boldsymbol{\beta}_l$), negative partition (-1 in $\boldsymbol{\beta}_l$), and ignored classes (0 in $\boldsymbol{\beta}_l$). Binary learning algorithms are then used to construct bit predictor h_l using training data

$$\mathbf{Z}_l = \{(\mathbf{x}_1, B_{y_1,l}), \ldots, (\mathbf{x}_m, B_{y_m,l})\} \tag{4.1}$$

with $B_{y_i,l} \neq 0$, for $l = 1, \ldots, L$ (throughout the rest of this chapter, we use "bit predictor" to denote the binary classifier associated with a column of the coding matrix). Clearly, classical multi-class categorization algorithms, such as *one-vs-one* and *one-vs-all* are special cases under the *ECOC* framework, with special choice of coding matrix [3]. Moreover, results in [3] suggest that learning a coding matrix in a problem-dependent way is better than using a pre-defined one. However, strong error-correcting ability alone does not guarantee good classification [31], since the performance of output coding is also highly dependent on the accuracy of the individual bit predictors. Consequently, several approaches [31, 51, 115] optimizing coding matrix and bit predictors simultaneously have been proposed. However, the coupling of learning coding matrix and bit predictors in a unified optimization framework is both a blessing and a curse. On the one

hand, it could directly assess the accuracy of each bit predictor and hence pick the coding matrix that avoids difficult bit prediction problems; on the other hand, simultaneous optimization often results in expensive computation, hindering these approaches from being applied to large-scale multi-class problems. Consequently, for the sake of scalability to massive number of classes, *SpOC* decouples the learning processes of code matrix and bit predictors. Therefore, the expensive procedure of learning bit predictors only needs to be carried out once, instead of multiple times in aforementioned approaches that learn code matrix and bit predictors simultaneously. However, our proposed approach still balances error-correcting ability of the code matrix and potential accuracy of associated bit predictors. Moreover, we also consider other properties that could affect classification accuracy of output coding based multi-class classifier, such as correlation among bit predictors, and complexity of each bit prediction problem. To the best of our knowledge, we provide the first attempt in learning optimal code matrix that explicitly considers multiple competing factors, and the fact that code matrix is learned without training associated binary classifiers multiple times enables our approach applicable to massive multi-class classification problems.

Given a test instance $\mathbf{x}$, the decoding procedure finds the class y whose codeword in $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_L(\mathbf{x}))$. For binary output coding scenario, where $\mathbf{B} \in \{-1, +1\}^{K \times L}$, either Hamming distance or Euclidean distance could be adopted to measure distance between two codewords. However, in the ternary case, where $\mathbf{B} \in \{-1, 0, +1\}^{K \times L}$, the special 0 symbol indicating ignored classes could raise problems. Specifically, previous attempts in decoding ternary codes [45] either (1) treat "0" bits the same way as non-zero bits, or (2) ignore those "0" bits entirely and only use non-zero bits for decoding. However, neither of the above approaches would prove sufficient. Specifically, treating "0" bits the same way as non-zero ones would introduce bias in decoding, since the distance increases with the number of positions that contain the zero symbol. On the other hand, ignoring "0" bits entirely would discard great amount of information. In our proposed framework, *probabilistic decoding* utilizes zero bits by propagating labels from non-zero bits to zero ones subject to smoothness constraints, and proves effective especially on large scale problems.

Our goal in this work is to design a multi-class classification method that is both accurate and fast when facing a large number of categories. Specifically, we propose *sparse output coding (SpOC)*, which turns the original large-scale K-class classification into an L-bit code construction problem, where $L = \mathcal{O}(\log(K))$ and each bit can be constructed in parallel through a binary off-the-shelf classifier; followed by a probabilistic decoding scheme to extract the class label.

Notations. Given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, we denote $\|\mathbf{A}\|_1 = \sum_{i,j} |A_{ij}|$ as its l_1 vector norm. For square matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$, $tr(\mathbf{A})$ is its trace. For $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, we denote $\mathbf{A} \odot \mathbf{B}$ as their Hadamard (a.k.a., element-wise) product. For $m \times n$ matrix $\mathbf{A}$ and $p \times q$ matrix $\mathbf{B}$, we denote $\mathbf{A} \otimes \mathbf{B}$ as their Kronecker product, which is an $mp \times nq$ matrix defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & \cdots & A_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & \cdots & A_{mn}\mathbf{B} \end{bmatrix} \tag{4.2}$$

For vector $\mathbf{x} \in \mathbb{R}^n$, its infinity norm is $\|\mathbf{x}\|_\infty = \max\{|x_1|, \ldots, |x_n|\}$. Moreover, for function $f(\mathbf{x})$, its conjugate function is $f^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathrm{dom} f}[\mathbf{y}^\top \mathbf{x} - f(\mathbf{x})]$. Also, $\|\mathbf{y}\|_* = \sup_{\|\mathbf{x}\| \leq 1} \mathbf{x}^\top \mathbf{y}$ is

the dual norm of $\|\cdot\|$. Finally, define $\mathrm{sign}(x)$ as the sign of x.

4.2 Coding

In this section, we provide details of the formulation for learning optimal code matrix for large-scale multi-class classification, together with efficient optimization algorithm.

4.2.1 Formulation

Output coding employs a code matrix to break a potentially massive multi-class problem into a series of binary bit predictions. Clearly, code matrix is crucial for the success of output coding, and its suitability could be measured using several competing factors, such as error-correcting ability, learnability of each bit predictor, and correlation between bit predictors.

As its most attractive advantage, the code matrix in output coding is usually chosen for strong error-correcting ability. That is to say, the optimal code matrix should have maximal separation between codewords for different classes. Besides codeword separation, since output coding is essentially aggregating discriminative information residing in each bit, learning accurate bit predictors is also crucial for its success. However, we usually do not know whether a binary partition can be well handled by the base bit predictor, unless a bit predictor has been learned on the partition. Unfortunately, the high computational cost associated with methods optimizing coding matrix and bit predictors simultaneously [51] renders them unfavorable in large-scale problems. To overcome this difficulty, we propose to use the training data and structure information among classes, to provide a measure of separability for each binary partition problem. Specifically, if some classes are closely related but are given different codes in the l-th bit, the bit predictor h_l may not be easily learnable. However, a binary partition is more likely to be well solved if the intra-partition similarity is large while the inter-partition similarity is small. Moreover, as output coding predicts class label by combining information from all bits, an ideal code matrix should have uncorrelated columns. Specifically, uncorrelated columns mean each bit predictor is focusing on a unique sub-problem of the original multi-class classification, while highly correlated columns severely limit the amount of information available at decoding. Finally, enforcing sparsity of the code matrix, i.e., introducing ignored classes in bit predictions, is crucial for massive multi-class classification. In this section, we will provide details for each of the aforementioned pieces, and formulate learning optimal code matrix as an orthogonality constrained optimization problem.

Before presenting detailed formulation for learning code matrix, we would like to make clear that the goal and contribution of this work is effective multi-class classification with massive number of classes, where any complex method would fail, and simplicity prevails. As a result, motivation for design of each piece in the optimization problem is a balance between effectiveness and efficiency. Although there might be more sophisticated formulations of the optimization problem, they will very likely increase computational cost, ultimately rendering the method incapable of handling massive multi-class classification.

Codeword Separation

Given an example $\mathbf{x}$, an L-dimensional bit predictor $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is computed. We then predict its label y based on which row in $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x})$. To increase tolerance of errors occurred in bit predictions, a crucial design objective of the code matrix is to ensure that the rows in $\mathbf{B}$ are separated as far from each other as possible. Hence, we propose to maximize the distance between rows in $\mathbf{B}$. Equivalently, we could minimize their inner products. Thus, codeword correlation of $\mathbf{B}$ could be computed as following:

$$\sum_{k=1}^{K}\sum_{k'=1}^{K}\mathbf{r}_k^\top\mathbf{r}_{k'} = \mathbf{e}_K^\top(\mathbf{B}\mathbf{B}^\top)\mathbf{e}_K = tr(\mathbf{B}^\top\mathbf{E}\mathbf{B}) \tag{4.3}$$

where $tr(\cdot)$ is matrix trace, $\mathbf{r}_1^\top, \ldots, \mathbf{r}_K^\top$ are row vectors of code matrix $\mathbf{B}$, $\mathbf{e}_K \in \mathbb{R}^K$ is the all-one vector and $\mathbf{E} = \mathbf{e}_K\mathbf{e}_K^\top$ is the $K \times K$ all-one matrix.

Learnability of Bit Predictors

One key property of optimal code matrix is to ensure that the resulting bit prediction problems could be accurately solved. The key motivation of our mathematical formulation is to compute the following two measures using semantic relatedness matrix $\mathbf{S}$ (defined later in this section) for each binary partition problem: intra-partition similarity, and inter-partition similarity. Specifically, in each binary partition problem, both positive partition and negative partition are composed of data points from multiple classes in the original problem. To encourage better separation, those classes composing the positive partition should be similar to each other. The similar argument goes for those classes composing the negative partition, but they should be dissimilar from the former set which composes the positive partition. Specifically, separability of the l-th binary partition problem could be measured as follows:

$$\sum_{B_{kl}B_{k'l}>0} S_{kk'} - \sum_{B_{kl}B_{k'l}<0} S_{kk'} = \sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kl}B_{k'l}S_{kk'} \tag{4.4}$$

It should be noted that the above defined measure should subtract $\sum_k S_{kk}$. However, as $\sum_k S_{kk}$ is constant and will not affect optimization of $\mathbf{B}$, we omit this step. Finally, learnability of all bit predictions could be measured as

$$\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kl}B_{k'l}S_{kk'} = \sum_{l=1}^{L}\mathbf{e}_K^\top\left[\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top \odot \mathbf{S}\right]\mathbf{e}_K = \mathbf{e}_K^\top\left[\mathbf{B}\mathbf{B}^\top \odot \mathbf{S}\right]\mathbf{e}_K = tr(\mathbf{B}\mathbf{B}^\top\mathbf{S}) = tr(\mathbf{B}^\top\mathbf{S}\mathbf{B}) \tag{4.5}$$

where $\odot$ is Hadamard (a.k.a., element-wise) product of two matrices, $\boldsymbol{\beta}_l$ is the l-th column of $\mathbf{B}$, and we have used the fact that $\mathbf{B}\mathbf{B}^\top = \sum_l \boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top$ and $\mathbf{e}^\top(\mathbf{A} \odot \mathbf{B})\mathbf{e} = tr(\mathbf{A}\mathbf{B})$.

Semantic Relatedness Matrix: $\mathbf{S}$ measures similarity between classes, using training data and structure information among classes. Let $\mathcal{X}_i = \{X_1^{(i)}, \ldots, X_{|\mathcal{X}_i|}^{(i)}\}$ and $\mathcal{X}_j = \{X_1^{(j)}, \ldots, X_{|\mathcal{X}_j|}^{(j)}\}$ be two classes from the multi-class problem. Several approaches have been proposed to measure similarity/distance between them, such as *Hausdorff distance*, *match kernel* [57, 98], divergence

between probability distributions estimated from $\mathcal{X}_i$ and $\mathcal{X}_j$ [105], or even classification accuracy of binary classifiers trained to separate the classes [9]. In this work, we use sum match kernel [57], and define data similarity between $\mathcal{X}_i$ and $\mathcal{X}_j$ as

$$S_{ij}^D = \frac{1}{|\mathcal{X}_i|}\frac{1}{|\mathcal{X}_j|}\sum_{p=1}^{|\mathcal{X}_i|}\sum_{q=1}^{|\mathcal{X}_j|} K_D(X_p^{(i)}, X_q^{(j)}) \tag{4.6}$$

where superscript D indicates that the similarity is estimated from data (in comparison to the similarity estimated from class structure discussed later), K_D is a Mercer kernel and in this work we use linear kernel.

Moreover, classes in massive multi-class problems are rarely organized in a flat fashion, but instead with a taxonomical structure [22, 35], such as a tree. Besides, algorithms for learning class structure have also been proposed [9, 151], although this is beyond the scope of this work. Given the taxonomical structure, we define structural affinity A_{ij} between class i and class j the same way as in Eq. (3.6), i.e.,

$$A_{ij} = \text{intersect}(P_i, P_j) / \max(\text{length}(P_i), \text{length}(P_j)) \tag{4.7}$$

where P_i is the path from root node to node i and $\text{intersect}(P_i, P_j)$ counts nodes shared by two paths P_i and P_j. We then construct structural similarity matrix

$$\mathbf{S}^S = \exp(-\kappa(\mathbf{E} - \mathbf{A})) \tag{4.8}$$

where κ is a constant controlling the decay factor. It should be noted that although we use class structure to define similarity, the goal of this work is not to propose yet another hierarchical classifier. In cases without such hierarchy, other ways of defining similarity between classes suffice as well (for example, we could simply use $\mathbf{S}^D$ only). Finally, semantic relatedness matrix $\mathbf{S}$ is the weighted sum

$$\mathbf{S} = \alpha\mathbf{S}^D + (1-\alpha)\mathbf{S}^S \tag{4.9}$$

with $\alpha \in [0, 1]$ being the weight.

Relaxation and Bit Correlation

Theoretical work [31] shows that learning discrete code matrix directly is NP-complete. Thus, we follow [31] and allow code matrix to take real values, followed by post-processing (taking the sign) to get the discrete code matrix.

Moreover, the power of output coding for multi-class classification stems from the fact that the final prediction is obtained by combining information from multiple bit predictors. Consequently, the more uncorrelated the bit predictors are, the more information we have at decoding time, and hence the better classification accuracy can be expected. As an extreme example, if all columns of the code matrix are solving the same binary separation problem, the amount of information available at decoding time is one single bit, and it is obviously not sufficient for accurate multi-class classification. Therefore, to ensure maximal amount of information at decoding, an ideal code matrix should have uncorrelated columns, such that each bit predictor is tackling a

unique sub-problem. To minimize bit correlation, we constrain the columns in code matrix to be orthogonal to each other, i.e.,

$$\mathbf{B}^\top \mathbf{B} = \mathbf{I} \tag{4.10}$$

where $\mathbf{I}$ is the identity matrix.

Sparse Code Matrix

For massive multi-class problems, it is crucial to introduce ignored classes, i.e., 0 in the code matrix [3, 116]. Otherwise, every bit predictor needs to consider the entire data. As an illustrating example, consider the ImageNet problem. With each of the 21841 classes participating in training a bit predictor, we will likely be facing a binary partition problem where both the positive and negative partitions are populated with data points coming from thousands of different classes. Clearly, learning bit predictor for such binary partition will be extremely difficult, due to the huge intra-partition dissimilarity. Therefore, to introduce ignored classes in bit predictors, we further regularize the l_1 norm of $\mathbf{B}$.

Final Formulation

Combining all pieces together, optimal code matrix should have minimal codeword correlation, maximal learnability of bit predictors, sufficient sparsity to reduce complexity of learning bit predictors, and orthogonal columns for uncorrelated bits. Weighing the above objectives, we propose the following formulation for learning optimal code matrix $\mathbf{B} \in \mathbb{R}^{K \times L}$

$$\min_{\mathbf{B}} \quad \frac{1}{2} tr[\mathbf{B}^\top (\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \lambda_1 \|\mathbf{B}\|_1 \tag{4.11}$$

$$s.t. \quad \mathbf{B}^\top \mathbf{B} = \mathbf{I} \tag{4.12}$$

where $\|\mathbf{B}\|_1 = \sum_{i,j} |B_{ij}|$ is its l_1 vector norm, λ_r and λ_1 are regularization parameters.
Selecting optimal parameters: It should be noted that unlike multi-class classification problems with small task space, the sheer size of the problem *SpOC* is designed to handle, makes it impossible to perform cross-validation or leave-one-out procedures to select optimal values for the parameters, such as α in (4.9), λ_r and λ_1 in (4.11). Thus, our approach for parameter selection is based on grid search, where we try several different values for each of $\{\alpha, \lambda_r, \lambda_1\}$ and solve problem (4.11) for optimal coding matrix. Then we compute the relative ratio among the three components in objective function of (4.11). Finally, the optimal parameters are selected as the group resulting in the relative ratio closest to 1. The motivation for such strategy is to ensure that each piece in the objective function has comparable value, such that all pieces could contribute and compete for optimal code matrix.

4.2.2 Optimization

The difficulty of solving problem (4.11) lies in the non-smoothness of the l_1 regularization on $\mathbf{B}$, and the orthogonality constraint (4.12). In this work, we employ *alternating direction method of*

multipliers (ADMM) [20] to effectively reduce the l_1 regularized problem into a series of l_2 regularized problems, where each problem is solved using gradient descent with Cayley transform to preserve orthogonality constraint on $\mathbf{B}$ and curvilinear search for optimal step size [135].

Alternating Direction Method of Multipliers

ADMM is a simple yet powerful algorithm, which takes the form of a decomposition-coordination procedure [20], where the solutions to small local subproblems are coordinated to find a solution to a large global problem. *ADMM* is first introduced in the 1970s [49], with most of the theoretical results established in the 1990s [42]. Moreover, it is shown in [20] that *ADMM* converges to local optimal point for non-convex optimization problems. However, until very recently, *ADMM* was not widely known in the computer vision or machine learning community. For completeness, we provide a brief review of the algorithm (for more details, see [20]). *ADMM* solves problems in the form

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}),\ s.t.\ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \tag{4.13}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times m}$, and $\mathbf{c} \in \mathbb{R}^p$. For problem (4.13), the augmented Lagrangian is formed as follows:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \tag{4.14}$$

where $\rho > 0$ is called the penalty parameter. *ADMM* consists of the following iterations [20]

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \tag{4.15}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \tag{4.16}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}) \tag{4.17}$$

To solve problem (4.11), we first reformulate it as follows

$$\min_{\mathbf{B},\mathbf{Z}} \quad \frac{1}{2} tr[\mathbf{B}^\top(\lambda_r\mathbf{E} - \mathbf{S})\mathbf{B}] + \mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) + \lambda_1\|\mathbf{Z}\|_1 \tag{4.18}$$

$$s.t. \quad \mathbf{B} - \mathbf{Z} = \mathbf{0} \tag{4.19}$$

where $\mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) = 0$ if constraint $\mathbf{B}^\top\mathbf{B} = \mathbf{I}$ is satisfied, and $\mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) = +\infty$ otherwise. Then *ADMM* solves problem (4.18) using the following iterations:

$$\mathbf{B}^{k+1} = \arg\min_{\mathbf{B}} \left(f(\mathbf{B}) + \frac{\rho}{2}\|\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k\|_2^2 \right) \tag{4.20}$$

$$\mathbf{Z}^{k+1} = \mathcal{S}_{\lambda_1/\rho}(\mathbf{B}^{k+1} + \mathbf{U}^k) \tag{4.21}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{B}^{k+1} - \mathbf{Z}^{k+1} \tag{4.22}$$

where $f(\mathbf{B})$ is defined as

$$f(\mathbf{B}) = tr[\mathbf{B}^\top(\lambda_r\mathbf{E} - \mathbf{S})\mathbf{B}] + \mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) \tag{4.23}$$

and $\mathcal{S}$ is the soft-thresholding operator defined as

$$\mathcal{S}_\kappa(a) = \max\{(1 - \kappa/|a|)a, 0\} \tag{4.24}$$

In the above *ADMM* iterations, both $\mathbf{Z}$ update (7.24) and $\mathbf{U}$ update (7.25) are trivial to compute. The $\mathbf{B}$ update in (7.23) is equivalent to the following constrained optimization

$$\min_{\mathbf{B}} \quad \frac{1}{2} tr[\mathbf{B}^\top(\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \frac{\rho}{2}\|\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k\|_2^2 \tag{4.25}$$

$$s.t. \quad \mathbf{B}^\top\mathbf{B} = \mathbf{I} \tag{4.26}$$

Comparing the above problem with (4.11), we can see that *ADMM* effectively reduces an l_1 regularized problem into a series of l_2 regularized problems.

Solving Problem (4.25) Using Cayley Transform and Curvilinear Search

Problem (4.25) is difficult to optimize due to the orthogonality constraint (4.26) on $\mathbf{B}$. In this work, we follow state-of-the-art technique [135], and solve problem (4.25) using gradient descent, with Cayley transform to preserve the orthogonality constraint and curvilinear search for optimal step size. In each iteration of the algorithm, given current feasible solution $\mathbf{B}$, the gradient of the objective function w.r.t. $\mathbf{B}$ could be computed as

$$\mathbf{G} = (\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B} + \rho(\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k) \tag{4.27}$$

Then a skew-symmetric matrix $\mathbf{A}$ is computed as

$$\mathbf{A} = \mathbf{G}\mathbf{B}^\top - \mathbf{B}\mathbf{G}^\top \tag{4.28}$$

The next new trial point $\mathbf{B}(\tau)$ is determined by the Crank-Nicolson like scheme [135]

$$\mathbf{B}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{B} \tag{4.29}$$

It is easy to verify that

$$\mathbf{B}(\tau)^\top\mathbf{B}(\tau) = \mathbf{B}^\top\mathbf{B} \tag{4.30}$$

i.e., every intermediate result is feasible, as long as initial point $\mathbf{B}$ satisfies the orthogonality constraint. For fast convergence, we adopt the Barzilai-Borwein step size in curvilinear search [135] to find optimal τ. Moreover, since $(\mathbf{I} + \frac{\tau}{2}\mathbf{A})^{-1}$ dominates the computation in (4.29), we apply the Sherman-Morrison-Woodbury theorem for efficient computation of matrix inverse. Theoretical results in [135] show the above algorithm converges to local optimal point.

Finally, we present in Algorithm 2 the method for learning optimal code matrix.

4.3 Probabilistic Decoding

For large-scale multi-class categorization, a sparse output coding matrix is necessary to ensure the learnability of each bit predictor. However, the zero bits in coding matrix also bring difficulty

Algorithm 2 Sparse Output Coding: Optimal Code Matrix Learning

Initialize $\mathbf{B}$ with randomly generated orthogonal matrix, $\mathbf{Z} = \mathbf{U} = \mathbf{0}$
repeat
  repeat
    Compute skew-symmetric matrix $\mathbf{A}$
    Curvilinear search for optimal step size τ
    Update new trial point $\mathbf{B}(\tau)$ as in Eq. (4.29)
  until stopping criterion satisfied [135]
  $\mathbf{Z}$ update using Eq. (7.24)
  $\mathbf{U}$ update using Eq. (7.25)
until stopping criterion satisfied [20]

Table 4.1: Decoding strategies for *error correcting output coding*.

DECODING ALGORITHM	OPTIMAL LABEL
HAMMING DECODING [39, 93]	$y^* = \arg\min_{y \in \mathcal{Y}} \sum_{l=1}^{L} \frac{1}{2}(1 - \text{sgn}(h_l(\mathbf{x}) \cdot B_{y,l}))$
EUCLIDEAN DECODING [107]	$y^* = \arg\min_{y \in \mathcal{Y}} \sqrt{\sum_{l=1}^{L}(h_l(\mathbf{x}) - B_{y,l})^2}$
ATTENUATED EUCLIDEAN DECODING [45]	$y^* = \arg\min_{y \in \mathcal{Y}} \sqrt{\sum_{l=1}^{L} \lvert B_{y,l} \rvert (h_l(\mathbf{x}) - B_{y,l})^2}$
LOSS-BASED DECODING [3]	$y^* = \arg\min_{y \in \mathcal{Y}} \sum_{l=1}^{L} loss(h_l(\mathbf{x}) B_{y,l})$
PROBABILITY-BASED DECODING [99]	$y^* = \arg\min_{y \in \mathcal{Y}} -\log\left(\prod_{l:B_{y,l} \neq 0} \mathbb{P}(h_l(\mathbf{x}) = B_{y,l}) + K\right)$

in decoding. Specifically, given an instance $\mathbf{x}$, we denote the vector of predictions generated by learned bit predictors as $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_L(\mathbf{x}))$. The decoding procedure in *output coding* is to find the class y for which codeword of $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x})$. In the simple case where a binary coding matrix $\mathbf{B} \in \{-1, +1\}^{K \times L}$ is adopted, the most frequently applied decoding approaches include Hamming decoding [39, 93] and Euclidean decoding [107], defined in Table 4.1. For ternary decoding with $\mathbf{B} \in \{-1, 0, +1\}^{K \times L}$, we could still apply those binary decoding strategies, treating 0 bits equally as non-zero ones, although we will encounter decoding bias as illustrated later in this section. One alternative strategy proposed in the literature ignores all zero bits in the coding matrix during decoding, and only counts the matching with non-zero bits. One example is attenuated Euclidean decoding [45], an adaptation of the Euclidean decoding strategy, which makes the measure unaffected by the zero bits of the codeword. Moreover, Allwein *et al.* [3] further improves the ternary decoding strategy by replacing the Euclidean distance with loss function, as defined in Table 4.1, where $h_l(\mathbf{x})B_{y,l}$ corresponds to the margin and $loss(\cdot)$ is a loss function that depends on the nature of the binary bit predictor. Finally, the authors of [99] propose a probability-based decoding strategy based on the continuous output of binary classifiers to deal with the ternary decoding. However, the probabilistic formulation in [99] only uses non-zero bits in the code matrix, and is thus equivalent to the loss-based decoding strategy in [3] with a logistic loss function. To sum up, although this latter group of approaches would avoid the problem of decoding bias on 0 bits of the coding matrix, it also discards significant amount of information, as only those non-zero bits are used for decoding.

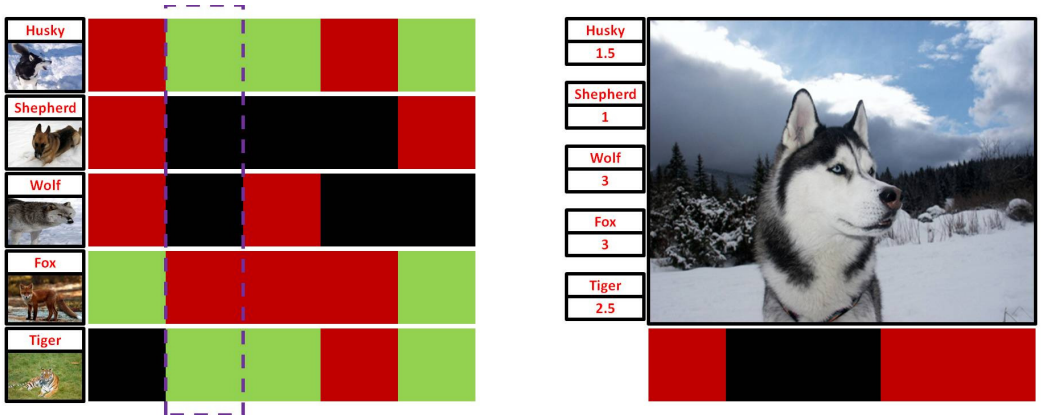

Figure 4.1: (Best viewed in color) Motivation for probabilistic decoding: (Left). one possible coding matrix for 5-class categorization, with $red = +1$, $black = -1$, and $green = 0$; (Right). one test image from class *Husky*, with its codeword shown in the bottom and Hamming distance with codewords for the 5 classes shown to the left. For the second bit (highlighted in dash box), although the first node (class *Husky*) is ignored during learning the bit predictor, it has a preference of being colored black, rather than red.

4.3.1 Motivating Example

As a motivating example, consider a 5-class problem in Figure 4.1. Given a test image from class *Husky*, if we treat zero bits the same way as non-zero ones, both Hamming decoding and Euclidean decoding would prefer *Shepherd* over *Husky*. However, *Husky* is only worse than *Shepherd* as its codeword has more zero. This effect occurs because the decoding value increases with the number of positions that contain the zero symbol and hence introduces bias. This problem might not seem severe in the example shown in Figure 4.1, however, for massive multi-class problems with large number of class labels, the bias introduced through zero bits would significantly affect classification accuracy. On the other hand, ignoring zero bits entirely would discard great amount of information that could potentially help in decoding the correct class label. This is especially true when K is large, where we expect a very sparse coding matrix to maximize learnability of each binary classifier. For example, in Figure 4.1, both classes *Husky* and *Tiger* have only two non-zero bits in their codewords. Since we cannot always have perfect bit predictors, classification errors on bit 1 and bit 4 would severely impair the overall accuracy. Therefore, it is our goal in this work to effectively utilize information residing in zero bits to effectively decode ternary output codes.

Fortunately, the semantic class similarity $\mathbf{S}$ computed using training data and class taxonomy, provides venue for effectively propagating information from non-zero bits to zero ones. For the example in Figure 4.1, class *Husky* is more similar to *(Shepherd, Wolf)* than *(Fox, Tiger)*. The second bit predictor in Figure 4.1 solves a binary partition of *(Shepherd, Wolf)* against *Fox*. Even though class *Husky* is ignored in training for this bit, the binary partition on images from this class will have a higher probability of being $+1$, due to the fact that the two positive classes in this binary problem are closely related to class *Husky*. Therefore, those classes with non-zero bits

in the coding matrix, should effectively propagate their label to those initially ignored classes. In this section, we propose probabilistic decoding, to effectively utilize semantic class similarity for better decoding. Specifically, we treat each bit prediction (without loss of generality, say, the l-th bit) as a label propagation [155] problem, where the labeled data corresponds to those classes whose codeword's l-th bit is non-zero, and unlabeled data corresponds to those whose l-th bit is zero. The goal of label propagation is to define a prior distribution indicating the probability of one class being classified as positive in the l-th binary partition. Combining this prior with the available training data, we formulate the decoding problem in *sparse output coding* as *maximum a posteriori* estimation.

4.3.2 Formulation

Given code matrix $\mathbf{B} \in \{-1, 0, +1\}^{K \times L}$, our decoding method estimates conditional probability of each class k given input $\mathbf{x}$ and L bit predictors $\{h_1, \ldots, h_L\}$. Without loss of generality, we assume the bit predictors constructed in the coding stage are linear classifiers, each parameterized by a vector $\mathbf{w}$ as $h_l(\mathbf{x}) = \text{sign}(\mathbf{w}_l^\top \mathbf{x})$. Define $(c_1, \ldots, c_L) \in \{-1, +1\}^L$ as a random vector of binary values, representing one possible codeword for instance $\mathbf{x}$. The decoding problem is then to find the class k, which maximizes the following conditional probability:

$$\begin{aligned}
\mathbb{P}(y = k|\mathbf{w}_1, \ldots, \mathbf{w}_L, \mathbf{x}, \boldsymbol{\mu}) &= \sum_{\{c_l\}} \mathbb{P}(y = k|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}, \{c_l\}) \cdot \mathbb{P}(\{c_l\}|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}) \\
&= \sum_{\{c_l\}} \mathbb{P}(y = k|\boldsymbol{\mu}, \{c_l\}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&\propto \sum_{\{c_l\}} \prod_l \mathbb{P}(c_l|y = k, \mu_{kl}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \sum_{\{c_l\}} \prod_l \mu_{kl}^{c_l} (1 - \mu_{kl})^{1 - c_l} \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \prod_l \{\mu_{kl} \mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}) + (1 - \mu_{kl})(1 - \mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}))\} \quad (4.31)
\end{aligned}$$

where $\{c_l\} = \{c_1, \ldots, c_L\}$, $\{\mathbf{w}_l\} = \{\mathbf{w}_1, \ldots, \mathbf{w}_L\}$, and $\mu_{kl} \in [0, 1]$ is the parameter in Bernoulli distribution $\mathbb{P}(c_l = 1|y = k) = \mu_{kl}$. Moreover, given the learned bit predictors, $\mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x})$ could be computed using a logistic link function as follows

$$\mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}_l^\top \mathbf{x})} \quad (4.32)$$

Therefore, in order to employ conditional probability in decoding for ternary output codes, we need to learn the values of Bernoulli parameters $\{\mu_{kl}\}_{k=1,\ldots,K}^{l=1,\ldots,L}$, which measures the probability of the l-th bit being $+1$ given the true class as $y = k$. Specifically, for the l-th column of the coding matrix, those classes corresponding to $+1$ in the l-th bit, i.e., $B_{kl} = 1$, will have $\mu_{kl} = 1$, and similarly those classes corresponding to -1, i.e., $B_{kl} = -1$, will have $\mu_{kl} = 0$. However, originally ignored classes (those corresponding to 0 in the coding matrix) will also be likely to have a preference on the value of the l-th bit. For the example in Figure 4.1, the second

bit predictor separates *(Shepherd, Wolf)* from *Fox*. Clearly, $\mathbb{P}(c_l = 1|Shepherd) = \mathbb{P}(c_l = 1|Wolf) = 0$ and $\mathbb{P}(c_l = 1|Fox) = 1$. Since class *Husky* is not directly involved in this binary classification problem, a non-informative prior would put $\mathbb{P}(c_l = 1|Husky) = 0.5$. However, if the true class for an instance $\mathbf{x}$ is *Husky*, this bit clearly has a much higher probability of being -1 than $+1$, due to the fact that *Husky* is much closer to *Shepherd* and *Wolf* semantically, than *Fox*. Therefore, we should have $\mathbb{P}(c_l = 1|Husky) < 0.5$, and in such way those classes with non-zero values in the l-th bit effectively propagate their label through the semantic class similarity $\mathbf{S}$ to those initially ignored classes.

Prior Distribution via Label Propagation

Following the motivating example in Figure 4.1, we define a prior distribution over Bernoulli parameters $\boldsymbol{\mu}$ such that labeled nodes effectively propagate their labels to those unlabeled nodes following the class hierarchy. Since each column $\boldsymbol{\beta}_l$ in the coding matrix will have its own labeling (different composition of positive classes, negative classes, and ignored classes) and label propagation for each column is independent of others, without loss of generality, we will focus on the l-th column. Suppose we have $\hat{l}$ classes participating in learning the l-th binary classifier, corresponding to the $\hat{l}$ non-zero terms in the l-th column $\boldsymbol{\beta}_l$ of coding matrix $\mathbf{B}$. Moreover, we also have $\hat{u} = K - \hat{l}$ ignored classes, corresponding to zeros in $\boldsymbol{\beta}_l$. Without loss of generality, we assume the first $\hat{l}$ classes are labeled as $(y_1, \ldots, y_{\hat{l}}) \in \{-1, +1\}^{\hat{l}}$. Consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with K nodes $\mathcal{V}$ corresponding to the K classes, where nodes $\hat{L} = \{1, \ldots, \hat{l}\}$ correspond to the labeled classes, and nodes $\hat{U} = \{\hat{l} + 1, \ldots, K\}$ correspond to ignored classes. Our task is to assign probabilities to nodes $\hat{U}$ being labeled as positive, using label information on nodes $\hat{L}$ and the graph structure of $\mathcal{G}$. Define $\boldsymbol{\mu}_l = (\mu_{1l}, \ldots, \mu_{Kl})$ as labels on nodes $\mathcal{V}$, where $\mu_{kl} = 1$ for those classes labeled as $+1$ and $\mu_{kl} = 0$ for those classes labeled as -1 in the l-th column of coding matrix $\mathbf{B}$. Consequently, the value of $\boldsymbol{\mu}_l$ on those unlabeled nodes represents our belief of it being labeled as $+1$. Equivalently, the distribution of $\boldsymbol{\mu}_l$ defines a prior on our Bernoulli parameters, in the sense that $\mu_{kl} = \mathbb{P}(c_l = 1|y = k)$. Intuitively, we want unlabeled nodes that are nearby in the graph to have similar labels, and this motivates the choice of the following quadratic energy function [155]:

$$E(\boldsymbol{\mu}_l) = \frac{1}{2} \sum_{i,j} S_{ij} (\mu_{il} - \mu_{jl})^2 \tag{4.33}$$

where $\mathbf{S}$ is the semantic similarity matrix. To assign probability distribution on $\boldsymbol{\mu}_l$, we form Gaussian field [155]

$$p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C} \exp(-CE(\boldsymbol{\mu}_l)) \tag{4.34}$$

where C is inverse temperature parameter, and

$$Z_C = \int_{\boldsymbol{\mu}_l | \forall k \in \hat{L}: \mu_{kl} = \frac{1}{2}(B_{kl}+1)} \exp(-CE(\boldsymbol{\mu}_l)) d\boldsymbol{\mu}_l \tag{4.35}$$

is a normalizing constant over all possible $\boldsymbol{\mu}_l$ constrained to $\boldsymbol{\beta}_l$ on non-zero terms, i.e., the labeled nodes in the graph corresponding to $\boldsymbol{\beta}_l$. Define diagonal degree matrix $\mathbf{D}$ with $D_{ii} = \sum_j S_{ij}$ and

graph Laplacian $\mathbf{\Delta} = \mathbf{D} - \mathbf{S}$, the Gaussian field defined on $\boldsymbol{\mu}$ could be equivalently formulated as follows:

$$p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C} \exp(-C\boldsymbol{\mu}_l^\top \mathbf{\Delta}\boldsymbol{\mu}_l) \tag{4.36}$$

with $\mu_{kl} = 1$ on classes labeled as $+1$ in $\boldsymbol{\beta}_l$ and $\mu_{kl} = 0$ on classes labeled as -1 in $\boldsymbol{\beta}_l$. Consequently, $p_C(\boldsymbol{\mu}_l)$ defines a prior distribution on $\boldsymbol{\mu}_l$, following the semantic class similarity, by clamping the labels on non-zero terms, and forcing smoothness of labels for zero terms, i.e., closely related classes should receive similar labels.

Parameter Learning

Given the L bit predictors learned in the coding stage of *sparse output coding*, and m training data points $\mathbf{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, we could calculate the conditional log-likelihood as follows:

$$\log \mathbb{P}(Y|\{\mathbf{w}_l\}, \mathbf{X}, \boldsymbol{\mu}) = \sum_{i=1}^{m}\sum_{l=1}^{L} \log\{\mu_{y_i l}\mathbb{P}_{li} + (1-\mu_{y_i l})(1-\mathbb{P}_{li})\} \tag{4.37}$$

where $\mathbb{P}_{li} = \mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}_i)$. Combining the above defined data likelihood with prior distribution over $\boldsymbol{\mu}$, we get the following optimization problem for learning parameters $\boldsymbol{\mu}$ using MAP estimation

$$\min_{\boldsymbol{\mu}} \quad -\sum_{i=1}^{m}\sum_{l=1}^{L} \log\{\mu_{y_i l}\mathbb{P}_{li} + (1-\mu_{y_i l})(1-\mathbb{P}_{li})\} + C\sum_{l=1}^{L} \boldsymbol{\mu}_l^\top \Delta\boldsymbol{\mu}_l \tag{4.38}$$

$$s.t. \quad 0 \le \mu_{kl} \le 1,\ k = 1, \ldots, K,\ l = 1, \ldots, L \tag{4.39}$$

$$\mu_{kl} = 1,\ \text{if } B_{kl} = +1 \tag{4.40}$$

$$\mu_{kl} = 0,\ \text{if } B_{kl} = -1 \tag{4.41}$$

where $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L]$. Clearly, $\boldsymbol{\mu}_l$ in the above optimization problem is independent of each other, and could therefore be optimized separately. We use projected gradient descent to solve the above optimization problem.

4.3.3 Decoding

Given the learned Bernoulli parameters $\boldsymbol{\mu}$, the inference targets to find the label k^* that maximizes the conditional probability:

$$k^* = \arg\max_k \mathbb{P}(y = k|\mathbf{w}_1, \ldots, \mathbf{w}_L, \mathbf{x}, \boldsymbol{\mu}) \tag{4.42}$$

Clearly, once the Bernoulli parameters $\boldsymbol{\mu}$ are obtained, decoding should take time that scales linearly with the number of columns in the coding matrix, which could be as small as $L = O(\log K)$. It should be noted that in order to enforce sparsity in the coding matrix, we usually pick $L = C\log K$, where C is a constant usually picked around 10 [3]. Still, our proposed probabilistic decoding is very efficient, especially when K is large, making it promising for large-scale multi-way classification. Moreover, besides decoding the most probable class label

Table 4.2: Data sets details.

DATA SET	#CLASS	#TRAIN	#TEST	#FEATURE
FLOWER	462	170K	170K	170006
FOOD	1308	467K	467K	170006
SUN	397	19850	19850	170006
IMAGENET	15952	2.5M	0.8M	338163

for an instance $\mathbf{x}$, the probabilistic decoding approach also naturally assigns confidence to each class, which will prove important when we not only want to generate a single label for instance $\mathbf{x}$ with potentially high risk, especially when the confidence gap between several classes is small.

4.4 Experiments

In this section, we test the performance of *sparse output coding* on two data sets: ImageNet [35] for object recognition, and SUN database [138] for scene recognition.

4.4.1 Data Sets and Feature Representations

Details of the data sets used in our experiments are provided in Table 4.2.

Object recognition on ImageNet. We start with two subtrees in ImageNet, with the root node being *"flower"* and *"food"*, respectively. The *flower* image collection contains a total of 0.34 million images covering 462 categories, and the *food* data set contains a total of 0.93 million images covering 1308 categories. For both data sets, we randomly pick 50% of images from each class as training data, and test on the remaining 50% images. Besides, we also carry out experiment on the entire ImageNet data. Specifically, we follow the same experimental protocols specified by [9, 136], where the data set is randomly split into 2.5 million images for training, 0.8 million for validation, and 0.8 million for testing, removing duplicates between training, validation and test sets by throwing away test examples which had too close a nearest neighbor in training or validation set [9].

Scene recognition on SUN database. The SUN database is by far the largest scene recognition data set, with 899 scene categories. We use 397 well-sampled categories to run the experiment [138]. For each class, 50 images are used for training and the other 50 for test.

Feature representations. For *flower*, *food* and *SUN* data sets, we employ the same feature representation for images as in [82]. Specifically, we compute SIFT [84] descriptors for each image, and then run *k-means* clustering on a random subset of 1 million SIFT descriptors to form a visual vocabulary of 8192 visual words. Using the learned vocabulary, we employ *Locality-constrained Linear Coding (LLC)* [82] for feature coding. Finally, a single feature vector is computed for each image using max pooling on a spatial pyramid [82]. Similar feature engineering is performed on the large *ImageNet* data set, but with a dictionary containing 16384 visual words, resulting in 338163 dimensional feature representation for each image.

Table 4.3: Flat error comparison on *flower*, *food* and *SUN* data sets.

	FLOWER (%)			FOOD (%)			SUN (%)		
ALGORITHM	TOP 1	TOP 5	TOP 10	TOP 1	TOP 5	TOP 10	TOP 1	TOP 5	TOP 10
OVR	72.77	39.95	27.43	75.02	46.59	34.23	83.38	69.72	61.83
RDOC	86.91	54.78	40.87	88.93	67.47	56.86	88.24	75.09	70.45
RSOC	87.12	53.69	39.04	86.52	66.88	57.45	88.11	75.12	71.83
SPECOC	78.63	47.75	35.91	81.94	58.12	45.70	85.91	72.63	64.38
SPOC-H	72.92	38.77	26.82	72.91	43.56	30.80	83.06	70.54	64.05
HIESVM-1	76.19	–	–	82.76	–	–	87.60	–	–
RELAXTREE	72.57	–	–	73.86	–	–	**81.09**	–	–
ECT	71.84	–	–	73.52	–	–	82.75	–	–
LET	73.48	40.21	28.13	74.18	45.03	32.46	83.69	69.78	61.97
SPOC	**69.73**	**34.35**	**24.08**	**70.98**	**43.28**	**29.00**	81.78	**68.65**	**61.26**

4.4.2 Experiment Design and Evaluation

We use *one-vs-rest (OVR)*, one of the most widely applied frameworks for multi-class classification, to serve as a baseline. It is interesting to compare against *OVR* since it is adopted as the major workhorse in several winning systems for multi-class object recognition competition [75, 82, 112]. Moreover, we also compare with three output coding based multi-class classification methods: (1) *random dense code output coding (RDOC)* proposed in [3] where each element in the code matrix is chosen at random from $\{-1, +1\}$, with probability $1/2$ for -1 and $+1$ each; (2) *random sparse code output coding (RSOC)* in [3], where each element in the code matrix is chosen at random from $\{-1, 0, +1\}$, with probability $1/2$ for 0, and probability $1/4$ for -1 and $+1$ each; (3) *spectral output coding (SpecOC)* proposed in [142], which builds dense output codes for multi-class classification, using spectral decomposition of the graph Laplacian constructed to measure class similarities. Moreover, to test the impact of *probabilistic decoding* on *SpOC*, we report results of *SpOC* using a simple Hamming distance based decoding strategy, denoted as *SpOC-H*. The third group of methods we compare with are hierarchical classifiers, a popular alternative to large-scale multi-class problems. Specifically, the first hierarchical classifier *(HieSVM-1)* follows a top-down approach, and trains a multi-class SVM at each node in the class hierarchy [68]. We have also tried a second hierarchical classification method *(HieSVM-2)* adopting the strategy in [33], where the hierarchical classifier is learned in a unified optimization framework. However, *(HieSVM-2)* runs into out of memory problems on all three data sets. Finally, we also compare against the *relaxed tree classifier (relaxTree)* [50], which learns relaxed tree hierarchy and corresponding classifiers in a unified framework via alternating optimization. Furthermore, we report results of two multi-class classification methods designed for problems with a large number of classes: (1) *error-correcting tournaments (ECT)* [12] which also reduces multi-class problem into a series of binary classifications; (2) *label embedding tree (LET)* [9] which learns a tree structure of classes by optimizing the overall tree loss, and performs multi-class classification via label embedding. Finally, for the largest data set in our empirical study, the ImageNet data set, we also report published accuracies to put our results into context (ideally, we would re-run those algorithms on our machines, however, due to the sheer size of the data,

such computation is very expensive). To ensure a fair comparison on the large *ImageNet* data, we also report the results of *SpOC* using the same features as adopted in [9], known as visual terms, a high-dimensional sparse vector of color and texture features. We denote the results of *SpOC* using visual terms for image features, as *SpOC-VT*.

For all algorithms except *label embedding tree* and *relaxed tree classifier*, we train *linear SVM* using *averaged stochastic gradient descent* [19]. For output coding based methods, we set code length $L = 200$ for *flower* and *SUN*, $L = 300$ for *food*, and $L = 1000$ for ImageNet. Data similarity matrix $\mathbf{S}^D$ is pre-computed with linear kernel and $\alpha = 0.5$. For *RDOC* and *RSOC*, 1000 random coding matrices are generated for each scenario and the one with the largest minimum pair-wise Hamming distances between all pairs of codewords and does not have any identical columns is chosen. To decode the label for *OVR* using learned binary classifiers, we pick the class with the largest decision value. For *RDOC*, *RSOC*, *SpecOC* and *SpOC-H*, we pick the class whose codeword has minimum Hamming distance with the codeword of test data point. Specifically, for decoding in *RSOC* and *SpOC-H*, we test both strategies of treating zero bits the same way as non-zero ones and ignoring zero bits entirely, and report the best result of these two methods. Finally, for *error-correcting tournaments (a.k.a. filter trees)*, we use the label tree structure learning method in [9] to learn a binary tree among classes, because we cannot use the given tree structure associated with data sets as *ECT* requires a binary tree.

Performance is measured using flat error and hierarchical error. For every data point, each algorithm will produce a list of 10 classes in the descending order of confidence (except *HieSVM-1*, *relaxTree* and *ECT*, which only provide the most confident class label), based on which the top-n flat error is computed, $n = 1, 5, 10$ in our case. Specifically, flat error equals 1 if the true class is not within the n most confident predictions, and 0 otherwise. On the other hand, hierarchical error reports the minimum height of the lowest common ancestors between true and predicted classes, using the given class hierarchical structure associated with the data sets. For each of the above two error measures, the overall error score for an algorithm is the average error over all test data points.

4.4.3 Results

Classification results for various algorithms are shown in Tables 4.3 to 4.5. For the ImageNet data, as the sheer size of the data renders it very expensive to run competing algorithms, we only provide accuracies of our method and *relaxTree* in Table 4.5. However, we also compare with the accuracies reported in [9] using *label embedding tree* and [136] using *learning to rank (L2R)*. Moreover, as feature engineering is crucial for image classification, we also report *ensemble* results in [136], combining multiple feature representations, such as spatial and multiscale color and texton histograms, and generating the final classification as an average of 10 separate models.

From results in Table 4.3 and Table 4.4, we have the following observations. (1) *SpOC* systematically outperforms *OVR*. More interestingly, *OVR* classifier consists of more than 1300 binary classifiers on *food* data set, while *SpOC* only involves 300 bit predictors. Although previous work has shown better results with *OVR* than output coding on small scale problems [112], with the number of classes increasing to thousands or tens of thousands, and hierarchical structure among classes, output coding with a carefully designed code matrix could outperform *OVR*, while maintaining cheaper computational cost, due to the error-correcting property introduced in

Table 4.4: Hierarchical error comparison on *flower*, *food* and *SUN* data sets.

	FLOWER	FOOD	SUN
OVR	1.95	3.26	1.15
RDOC	2.35	4.39	1.23
RSOC	2.38	4.10	1.24
SPECOC	2.27	4.02	1.21
SPOC-H	1.99	2.97	1.15
HIESVM-1	2.34	3.42	1.28
RELAXTREE	1.89	3.04	**1.02**
ECT	1.87	**2.89**	1.12
LET	1.98	3.09	1.16
SPOC	**1.69**	2.92	1.06

the code matrix. (2) Both *SpOC* and *OVR* beat *RDOC* and *RSOC*, revealing the importance of enforcing learnability of each bit predictor, since randomly generated code matrix could very likely generate difficult binary separation problems. (3) *SpOC* performs better than *SpecOC*, which employs a dense code matrix. The margin between *SpOC* and *SpecOC* is even more severe on *food*, revealing the importance of having ignored classes in each bit predictor. (4) *SpOC-H* generates inferior results than *SpOC* across the board, indicating the necessity of *probabilistic decoding* to handle zero bits in the code matrix. (5) *SpOC* and *OVR* both outperform *HieSVM-1*, where errors made in the higher level of the class hierarchy get propagated into the lower levels, with no mechanism to correct those early errors. However, the error-correcting property in *SpOC* introduces robustness to errors made in bit predictors. Results for *HieSVM-2* are not available as it runs into out of memory problems on all three data sets. (6) *SpOC* is comparable with *relaxTree* on *SUN* data set, but outperforms it on the other two data sets. Though *relaxTree* defers the decision on some difficult classes to lower level nodes, hence achieving better accuracy than conventional hierarchical classifier such as *HieSVM-1*, it still lacks the robustness or error correcting ability introduced by the coding matrix in *SpOC*. More interestingly, *relaxTree* involves multiple iterations of learning base classifiers due to the adoption of alternating optimization, and we will compare its time complexity against *SpOC* later this section. (7) *SpOC* beats both *ECT* and *LET*, both of which involve expensive procedure of learning optimal semantic class structure from data, while *SpOC* simply utilizes the class structure associated with the data set.

According to results in Table 4.5 for the large *ImageNet* data set, we see that *SpOC* clearly outperforms *OVR*, consistent with results reported on other data sets. It is interesting that *SpOC* also clearly beats *relaxTree*, revealing the importance of error correcting ability in real large-scale multi-class problems. Moreover, comparison with numbers reported in [9] and [136] shows that *SpOC* also outperforms *label embedding tree* and *learning to rank* on the large ImageNet task. Also, using the same features as [9], *SpOC-VT* clearly beats label embedding tree [9] with a significant margin, revealing the effectiveness of our proposed sparse output coding approach. Finally, *SpOC* result is comparable with the accuracy of *ensemble* method, where classification is obtained by combining various kinds of features and averaging over 10 separate models.

Finally, we visualize the coding matrix learned for the large *ImageNet* data set using our

Table 4.5: Classification accuracy on ImageNet (accuracy of *approximate kNN* is reported by [136]).

	ACCURACY
OVR	2.27%
APPROXIMATE KNN	1.55%
LET [9]	2.54%
L2R [136]	6.14%
ENSEMBLE [136]	10.03%
RELAXTREE	5.28%
SPOC-VT	9.15%
SPOC	**9.46%**

proposed algorithm. Due to the sheer size of the task space, we cannot show the entire coding matrix or the entire bit predictor, as each bit predictor usually involves thousands of original classes. Consequently, we randomly selected 6 bit predictors and show randomly picked classes from both positive and negative partitions. Specifically, Figure 4.2 shows the composition of positive partition and negative partition for several bit predictors, and we could already see the similarity within each partition, and distinction between the two partitions.

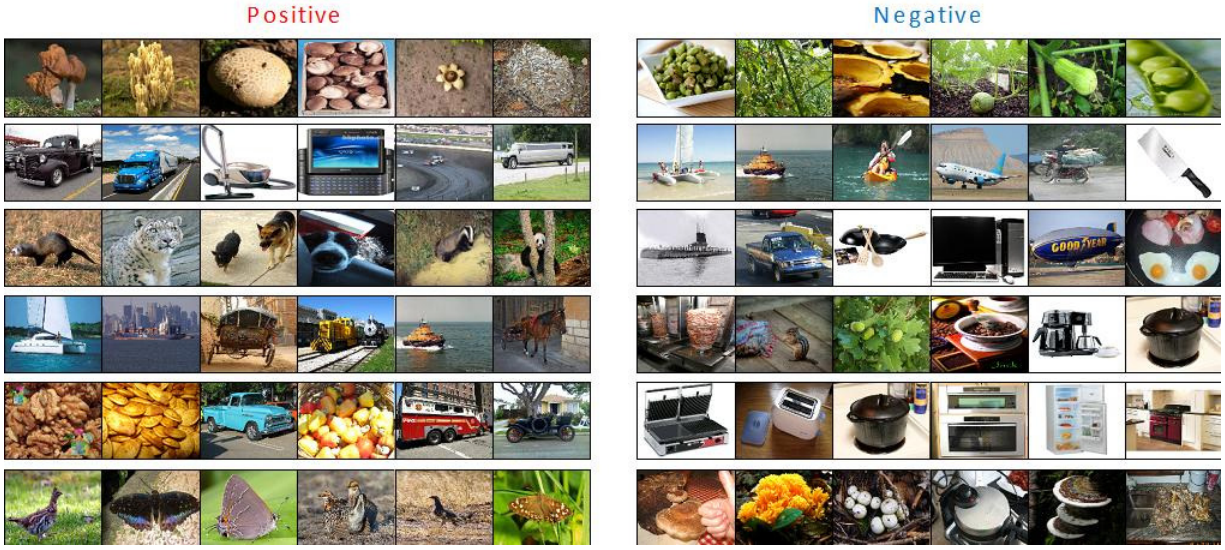

Figure 4.2: Visualization of 6 bit prediction problems generated by the learned coding matrix for *ImageNet* with $L = 1000$. Each row corresponds to a binary problem, with the left panel showing categories composing the positive partition, and right panel showing categories composing the negative partition.

Remarks on how to interpret our ImageNet result. It is widely acknowledged in the vision community [75, 82] that feature engineering is an important alternative source for boosting

Table 4.6: Classification error (flat error) and time complexity (seconds) of SpOC with various code lengths. T_c is the time for learning coding matrix and decoding, and T_b is the time for learning bit predictors. ($1E7 = 1 \times 10^7$)

	$L = 100$	$L = 200$	$L = 300$	$L = 400$
Top 1 (%)	74.71	69.73	68.82	68.79
Top 5 (%)	41.28	34.35	33.70	33.82
Top 10 (%)	30.12	24.08	22.83	22.76
T_c (seconds)	106.4	175.3	237.1	398.6
T_b (seconds)	1.72E7	3.24E7	4.89E7	6.49E7

image classification accuracy. Recently, with a standard *OVR* classifier system but sophisticated feature engineering that learns feature representations using deep network with 1 billion trainable parameters, strong results surpassing what we report here have been published [75]. However, it should be noted that our work focuses on techniques of training superior massive multi-class classifiers under any given features, and the work of designing state-of-the-art feature representation for images, should not be be viewed as competing efforts, but rather, complementary. Indeed we expect that one can directly apply *SpOC* on top of a superior feature representations learned in previous works to yield even better results. In fact, applying *SpOC* on feature representations learned in [82] already beats most state-of-the-art results on ImageNet data set. The superior result in [75] using deep learning for feature design should not discredit *SpOC*'s value as a principled way for massive multi-class classification, as the two approaches focus on different stages in the image classification pipeline. We consider combining *SpOC* with feature representations learned in [75] as an interesting future work once that feature representations on ImageNet are made publicly available.

4.4.4 Effect of Code Length

In this section, we investigate the effect of code length on classification accuracy of *SpOC*. Specifically, we test *SpOC* on the *flower* data set with various code lengths and report classification errors in Table 4.6. According to Table 4.6, classification error of *SpOC* decreases as the code length increases, as stronger error-correcting ability is accompanied with longer codes. However, the fact that $L = 200$ performs almost as well as $L = 400$ demonstrates that *SpOC* usually requires much less bit predictors compared to the number of classes in the multi-class classification problem.

4.4.5 Time Complexity

We also report computational time of *SpOC* on the *flower* data set with various code lengths in Table 4.6. Specifically, computational time for *SpOC* consists of three parts: (1) time for learning output coding matrix, (2) time for training bit predictors, and (3) time for probabilistic decoding. We implement *SpOC* using MATLAB 7.12 on a 3.40 GHZ Intel i7 PC with 16.0 GB main memory. Bit predictors are trained in parallel on a cluster composed of 200 nodes. Time for

Table 4.7: Time complexity comparison (seconds).

	FLOWER	FOOD	SUN
OVR	$1.68E8$	$2.63E8$	$1.71E7$
ECT	$5.60E7$	$8.07E7$	$4.28E6$
LET	$5.26E7$	$8.02E7$	$4.15E6$
RELAXTREE	$1.30E8$	$2.41E8$	$1.09E7$
SPOC	$3.24E7$	$5.02E7$	$3.72E6$

training bit predictors is the summation of time spent on each node of the cluster. According to Table 4.6, time for learning code matrix and probabilistic decoding is almost negligible compared to the time spent on training bit predictors.

Moreover, we also compare the time complexity of *SpOC* with that of *OVR*, *ECT*, *LET* and *relaxTree*. According to Table 4.7, the total CPU time of *SpOC* is systematically shorter than *OVR*, which is expected as *SpOC* requires training much less binary classifiers than *OVR*, and each bit predictor in *SpOC* only involves a subset of classes from the original problem, while each binary classifier in *OVR* is trained using data from all classes in the multi-class problem. This again reveals the advantage of *SpOC* over the widely popular *OVR* on massive multi-class classification. Moreover, *SpOC* is also more efficient than *ECT* and *LET*. One possible reason could be the expensive tree learning procedure involved in both *ECT* and *LET*, while the time spent on learning optimal code matrix in *SpOC* is negligible compared to the time of training bit predictors. Finally, *SpOC* clearly takes less computational time than *relaxTree*, which requires multiple iterations of learning base classifiers due to the alternating optimization approach in learning relaxed tree classifier.

4.5 Summary

To conclude this chapter, we summarize our main contributions as follows. (1) We propose an approach for large-scale visual recognition, with scalability to problems with tens of thousands of classes. *SpOC* is robust to errors in bit predictors, simple to parallel, and its computational time scales sub-linearly with the number of classes. (2) We propose efficient optimization based on *alternating direction method of multipliers*, where each sub-problem is solved using gradient descent with Cayley transform to preserve orthogonality constraint and curvilinear search for optimal step size. (3) We propose *probabilistic decoding* to effectively utilize semantic similarity between visual categories for accurate decoding. (4). We provide promising empirical results, tested on ImageNet with around $16,000$ classes. The fact that *SpOC* takes less bit predictors than *one-vs-rest* multi-class classification while achieving better accuracy, renders our proposed approach especially promising when scaling up to human cognition level multi-class classification.

Part II

Large Scale Video Understanding

Part II Large Scale Video Understanding

Now we turn our attention from categorizing large collection of image data to understanding temporally long or even endless video sequences. As is often the case, one of the major difficulties in video analysis is the huge amount of data, while it is often true that only a small portion of video contains important information. Consequently, algorithms that could automatically detect unusual events within streaming or archival video, or generate short summary for the originally long video, would significantly improve the efficiency of video analysis and save valuable human attention or concentrate downstream processing (such as various tasks of detection, action recognition, etc.) on only the most salient contents. Unusual event detection and automatic video summarization are the keys to unlock the huge amount of video sequences.

This part consists of three chapters. First, we provide a framework of using *sparse coding* and *online re-constructibility* to detect unusual events in videos. A query video segment is projected onto a set of sparse coding bases conceptually constituting usual events, which are learned and updated realtime by the algorithm, where the reconstruction error is obtained. An unusual event in a video is detected as those segments whose reconstruction errors are significantly higher than the majority of the other (usual event) segments of the video. To our knowledge, we offer the first treatment of unusual event detection in this sparse re-constructibility framework.

Second, we develop a method that offers the following function and alike: *"I only have 1 minute for this hour-long video, tell me where/what to watch"*. That is, it automatically compiles the most salient and informative portion of the video for users, by automatically scanning through video stream, in an online fashion, to remove repetitive and uninteresting contents. The summary generated by our proposed method is a short video itself, revealing the essence of the original video, just like a "trailer".

Third, we address the challenging problem in video summarization regarding the gap between the automated characterization of the important video segments and the human perception about the importance of a segment, since purely unsupervised video summarization could potentially miss some of these key components, or incorporate unimportant events. Specifically, within a class of videos of similar nature, such as a collection of wedding videos, users are asked to generate summaries for a subset of this class of videos. Given such manually generated summaries, our proposed method aims to learn the preferred storyline within the given class of videos. Then, it will automatically generate summaries for the rest of videos in the class, capturing the similar storyline in those manually summarized videos.

Chapter 5

Online Detection of Unusual Events in Videos via Dynamic Sparse Coding

One of the major difficulties in video analysis is the huge amount of data. However, it is often true that only a small portion of video contains important information. Consequently, algorithms that could automatically detect unusual events within streaming or archival video would significantly improve the efficiency of video analysis and save valuable human attention for only the most salient contents.

5.1 Introduction

In this chapter, we provide a framework of using *sparse coding* [77] and *online re-constructibility* to detect unusual events in videos. A query video segment is projected onto a set of sparse coding bases conceptually constituting usual events, which are learned and updated realtime by the algorithm, where the reconstruction error is obtained. An unusual event in a video refers to those segments whose reconstruction errors are significantly higher than the majority of the other (usual event) segments of the video. To our knowledge, we offer the first treatment of unusual event detection in this sparse re-constructibility framework. Compared to previous work that are either model-based [65, 80, 133], or clustering or saliency based [17, 55, 150], our proposed dynamic sparse coding framework is built upon a rigorous statistical principle, offering the following advantages: 1) It makes no prior assumptions of what unusual events may look like, hence no need to obtain prior models, templates, knowledge of the clusters; 2) It is completely unsupervised, leveraging only on the assumption that an unusual event is unlikely to occur in the small initial portion of a video; and 3) Our learning algorithm continues to learn and updates its bases dictionary as the algorithm observes more data, avoiding any issues with concept drift.

Figure 5.1 provides a flowchart of the proposed unusual event detection approach. Specifically, given a video sequence, the proposed method employs a sliding window along both the spatial and temporal axes to define an event. As the sliding window scans along the spatial and temporal axes, the video is broken into a set of events, each represented by a group of spatio-temporal cuboids. The task of unusual event detection is therefore formulated as detecting unusual group of cuboids residing in the same sliding window. A dictionary is first learnt from the

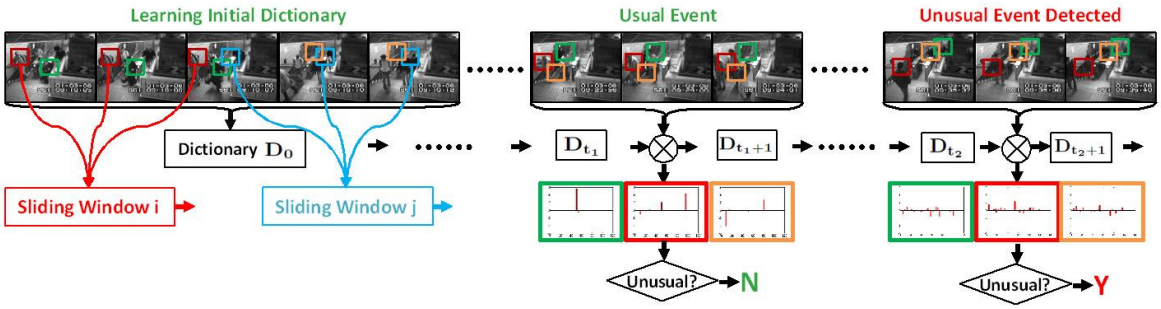

Figure 5.1: (Best viewed in color) Flowchart of our approach. Given an input video sequence, events are defined using sliding windows (displayed as colored boxes on the video frames). Within each sliding window, spatio-temporal interest points are detected (not shown in the figure), and a dictionary is learned using previously seen video data. For a query event, reconstruction vectors using bases in the dictionary are learned by solving a sparse coding optimization problem. Typicality of the query event is then decided using these vectors. Finally, the dictionary is updated with the addition of the query event.

video using sparse coding and later updated in an online fashion as more data become available. Given the learned dictionary, a reconstruction weight vector is learned for each query event and a typicality measure is computed from the reconstruction vectors. The proposed algorithm only needs to scan through the video once, and online updating of the learned dictionary makes the algorithm capable of handling concept drift in the video sequence. Finally, using sparse coding enables the algorithm to robustly discriminate between truly unusual events and noisy usual events.

It should be noted that the definition of unusual events is rather subjective. In this thesis, we define unusual events as those incidences that occur very rarely in the entire video sequence [1, 17, 54, 80, 133, 150].

5.2 Sparse Coding for Unusual Event Detection

In this section, we provide details of the proposed unsupervised algorithm for unusual event detection.

5.2.1 Video Representation

The proposed unusual event detection algorithm adopts a representation based on spatio-temporal cuboids (though it should be noted that the proposed approach could be applied over a variety of video descriptors), to detecte salient points within the video and describe the local spatio-temporal patch around the detected interest points. There have been several attempts in detecting spatio-temporal interest points in video sequences [15, 40, 73]. Here, we adopt the spatio-temporal interest points detected using the method in [40], and describe each detected

Figure 5.2: Example spatio-temporal interest points detected with the method in [40].

interest point with histogram of gradient (HoG) and histogram of optical flow (HoF). This spatio-temporal interest point based feature representation will be used throughout the rest of this thesis. Figure 5.2 provides several frames from the video data used in this chapter and the detected spatio-temporal interest points within these frames.

5.2.2 The Proposed Method

Given a video sequence, the proposed approach employs a sliding window along both the spatial and temporal axes to define an event. Consequently, as a video is represented as a set of cuboids, those cuboids residing in a sliding window define an event. As the sliding window scans along the spatial and temporal axes, the video is broken into a set of events, each represented by a group of spatio-temporal cuboids. Specifically, the video is represented as $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, with each event $\mathbf{X}_i$ composed of a group of cuboids, i.e., $\mathbf{X}_i = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{n_i}\}$, where n_i is the total number of cuboids within the sliding window.

A Sparse Coding Formulation

In this work, detecting unusual events in video is formulated as a sparse coding problem. The basic idea for our approach is to represent the knowledge of usual events using the learned dictionary $\mathbf{D}$, whose columns are bases for reconstructing signals. Different from conventional settings of sparse coding, where the input signal is a vector, the input signal in unusual event detection is an event, composed of a group of cuboids $\mathbf{X}_i = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{n_i}\}$. Therefore, the basic unit of input signal is no longer a vector, but instead a group of vectors, with both spatial and temporal location information. In addition to sparsity of the reconstruction weight vectors, we also need to consider the relationships between these weight vectors imposed by the neighborhood structure of cuboids that define the event.

Given dictionary $\mathbf{D}$ (details about learning $\mathbf{D}$ will be provided later in this section), we define the following objective function that measures the typicality of an event $\mathbf{X}_i = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{n_i}\}$ and a specific choice of reconstruction weight vectors $\boldsymbol{\alpha}_i = \{\boldsymbol{\alpha}_i^1, \ldots, \boldsymbol{\alpha}_i^{n_i}\}$:

$$J(\mathbf{X}_i, \boldsymbol{\alpha}_i, \mathbf{D}) = \frac{1}{2} \sum_j ||\mathbf{X}_i^j - \mathbf{D}\boldsymbol{\alpha}_i^j||_2^2 + \lambda_1 \sum_j ||\boldsymbol{\alpha}^j||_1 + \lambda_2 \sum_{j,k} \mathbf{W}_{jk} ||\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_i^k||_2^2 \qquad (5.1)$$

where subscripts j and k run through $\{1, \ldots, n_i\}$ and λ_1, λ_2 are regularization parameters. We now discuss in details of each term in Eq. (5.1).

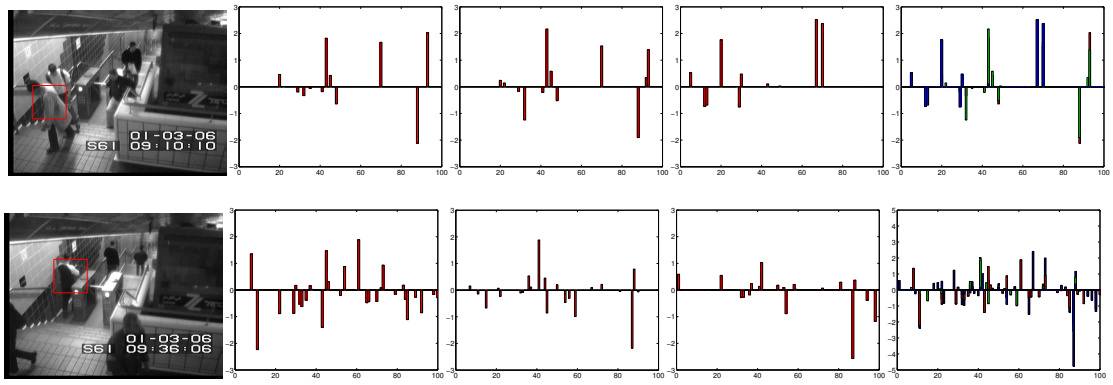

Figure 5.3: First row: usual event (leaving subway exit); second row: unusual event (entering subway exit). From left to right: example frame and sliding window, reconstruction vectors for 3 cuboids, plot containing all 3 reconstruction vectors on the same figure.

Reconstruction error. The first term in Eq. (5.1) is the reconstruction error. For a usual event, this term should be small, due to the assumption that the learned dictionary represents knowledge in the previously seen video data. A small reconstruction error means the information within the newly observed event $\mathbf{X}_i$ has appeared in early part of the video, which agrees with our definition of usual events.

Sparsity regularization. The second term is the sparsity regularization. Enforcing sparsity for reconstructing usual events is necessary due to the fact that dictionary $\mathbf{D}$ is learned to maximize the sparsity[1] of reconstruction vectors for usual events in the video. On the other hand, for unusual events, although it is possible that a fairly small reconstruction error could be achieved, we would expect using a large amount of video fragments for this reconstruction, resulting in a dense reconstruction weight vector. Figure 5.3 presents the reconstruction weight vectors for 2 events in the video: the first event is usual, and the second is unusual. Results in Figure 5.3 show that the reconstruction vectors for usual event are sparse, while the ones for unusual event are dense.

Smoothness regularization. The third term is the smoothness regularization, where $\mathbf{W} \in \mathbb{R}^{n_1 \times n_1}$ is the adjacency matrix of $\{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{n_i}\}$, with large value corresponding to neighboring cuboids and small value corresponding to far apart cuboids. This regularization is based on the fact that similar motions at neighboring patches are more likely to be involved in a usual event. Consequently, it should be of higher probability for similar reconstruction weight vectors being assigned to neighboring cuboids in a usual event. The adjacency matrix $\mathbf{W}$ adopted in this chapter is the Gaussian RBF kernel function:

$$W_{jk} = \exp\left[-\frac{||x_j - x_k||^2}{2\sigma^2} - \frac{||y_j - y_k||^2}{2\sigma^2} - \frac{||t_j - t_k||^2}{2\tau^2}\right] \tag{5.2}$$

where (x_j, y_j) and t_j are spatial and temporal locations of the jth cuboid, σ and τ are variances of the Gaussian function. In the last column of Figure 5.3, where all 3 reconstruction vectors

[1] In this chapter, we define sparsity as the number of zero elements in a vector.

are plotted on the same image, usual event shows a significant amount of overlap, while the reconstruction vectors for unusual event becomes even denser.

In summary, our sparse coding scheme presented above encapsulates the following intuitions for what we would think of usual and unusual events. Given a dictionary of bases corresponding to usual events, a usual event should be reconstructible from a small number of such bases, in a way that the reconstruction weights change smoothly over space/time across actions in such events. On the other hand, an unusual event is either not reconstructible from the dictionary of usual events with small error, or, even if it is reconstructible, it would necessarily build on a combination of a large number of bases in the dictionary, and possibly in a temporal-spatially non-smooth fashion. Crucial to this technique, is the ability to learn a good dictionary of bases representing usual events, and being able to update the dictionary online to adapt to changing content of the video, which we discuss in detail in next section.

Different from conventional sparse coding, where the bases in dictionary are fixed after training, the dictionary in our dynamic sparse coding framework is updated online to adapt to changing content of the video.

Optimization

The objective function $J(\mathbf{X}_i, \boldsymbol{\alpha}_i, \mathbf{D})$ of Eq. (5.1) measures the typicality of event $\mathbf{X}_i$ with any reconstruction weight vector $\boldsymbol{\alpha}_i$ and any dictionary $\mathbf{D}$. The lower J is, the more likely an event $\mathbf{X}_i$ is normal. As both $\boldsymbol{\alpha}_i$ and $\mathbf{D}$ are latent variables introduced in the formulation, to properly measure the typicality of an event $\mathbf{X}_i$, we need to adopt the optimal weight vector $\boldsymbol{\alpha}_i^*$ and dictionary $\mathbf{D}^*$ which minimize the objective function for the given event $\mathbf{X}_i$. Specifically, assume there are m events in the video defined using the sliding window, i.e., $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, the optimal reconstruction weight vector $\boldsymbol{\alpha}_i^*$ and dictionary $\mathbf{D}^*$ are learned by solving the following optimization problem

$$(\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_m^*, \mathbf{D}^*) = \underset{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m, \mathbf{D}}{\arg\min} \sum_{i=1}^{m} J(\mathbf{X}_i, \boldsymbol{\alpha}_i, \mathbf{D}) \tag{5.3}$$

subject to proper constraints discussed later. A close look into the above optimization problem reveals that the problem is convex with respect to the coefficients $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m\}$ of the sparse decomposition when the dictionary $\mathbf{D}$ is fixed, and also convex with respect to $\mathbf{D}$ when $\boldsymbol{\alpha}$ is fixed. However, it is not jointly convex with respect to $\mathbf{D}$ and $\boldsymbol{\alpha}$. A natural solution is to alternate between these two variables, minimizing one while clamping the other. We note that this alternating optimization algorithm converges to local optimum. With the learned dictionary $\mathbf{D}^*$, given a newly observed event $\mathbf{X}'$, the algorithm learns the optimal reconstruction weight vector $\boldsymbol{\alpha}'$ for this event. Consequently, $J(\mathbf{X}', \boldsymbol{\alpha}', \mathbf{D}^*)$ measures the typicality of event $\mathbf{X}'$. An event $\mathbf{X}'$ is detected as unusual if its corresponding $J(\mathbf{X}', \boldsymbol{\alpha}', \mathbf{D}^*)$ is larger than certain threshold.

Learning Reconstruction Weight Vector $(\boldsymbol{\alpha})$ with Fixed $\mathbf{D}$. With dictionary $\mathbf{D}$ fixed, reconstruction weight vectors for different events are independent. Therefore, they could be optimized independently. Specifically, for event $\mathbf{X}_i = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{n_i}\}$, the corresponding opti-

mization problem is as follows

$$\min_{\boldsymbol{\alpha}_i^1,\ldots,\boldsymbol{\alpha}_i^{n_i}} \frac{1}{2}\sum_j ||\mathbf{X}_i^j - \mathbf{D}\boldsymbol{\alpha}_i^j||_2^2 + \lambda_1 \sum_j ||\boldsymbol{\alpha}^j||_1 + \lambda_2 \sum_{j,k} \mathbf{W}_{jk}||\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_i^k||_2^2 \tag{5.4}$$

Except for the second term, both two other terms in the objective function are convex quadratic functions of $\boldsymbol{\alpha}_i$. For the above L_1 regularized convex function, the objective is not continuously differentiable. Consequently, the most straightforward gradient-based methods are difficult to apply [77]. Various approaches have been proposed to solve this problem: generic QP solvers (e.g., CVX), interior point method [26], a modification of least angle regression (LARS) [43] and grafting [102]. In this chapter, we adopt the feature-sign search algorithm introduced in [77] to solve the above L_1 regularized optimization method.

Learning Dictionary $(\mathbf{D})$ with Fixed $\boldsymbol{\alpha}$. With fixed coefficients $\boldsymbol{\alpha}$, the optimization problem for dictionary $\mathbf{D}$ is as follows

$$\min_{\mathbf{D}} \quad \frac{1}{2m}\sum_{i=1}^{m}\sum_{j=1,\ldots,n_i} ||\mathbf{X}_i^j - \mathbf{D}\boldsymbol{\alpha}_i^j||_2^2 \tag{5.5}$$

$$s.t. \quad \mathbf{D} \in \mathbb{R}^{m\times k} \tag{5.6}$$

$$\forall j = 1,\ldots,k,\ \mathbf{d}_j^T\mathbf{d}_j \leq 1 \tag{5.7}$$

The constraint in (5.7) is introduced to prevent terms in $\mathbf{D}$ from being arbitrarily large, which would result in arbitrarily small values of $\boldsymbol{\alpha}$ [77]. The above optimization problem is a least squares problem with quadratic constraints. In this work, we solve this problem using Lagrange dual.

5.2.3 Online Dictionary Update

As we stated in Section 5.1, one contribution of our work is to automatically learn the video dictionary and perform ongoing learning as we continue to observe the sequence. Unlike previous work where a model for usual events is first learned using training data [1, 17, 65], our fully unsupervised framework can be much more practical in real-world scenarios.

Specifically, the above formulation needs initial training data to learn the dictionary. In video surveillance, it is often challenging to obtain such suitable training data. Even if we were provided with a set of training data, we postulate that the bases dictionary learned from the training data is not necessarily optimal for detecting unusual events in new query videos. We therefore propose an online dictionary learning algorithm in this section that requires no training data other than the video sequence itself. Our idea is to first learn an initial dictionary using an initial portion of the video, and update this learned dictionary using each newly observed event.

Assume the algorithm has observed t-th event in the video, the optimal dictionary is the solution of the following optimization problem

$$\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{2t}\sum_{i=1}^{t}\sum_{j=1,\ldots,n_i} ||\mathbf{X}_i^j - \mathbf{D}\boldsymbol{\alpha}_i^j||_2^2 \tag{5.8}$$

where $\mathcal{C} = \{\mathbf{D} \in \mathbb{R}^{m \times k} : \ \mathbf{d}_j^T \mathbf{d}_j \leq 1, \ \forall j = 1, \ldots, k\}$. Ideally, to solve this problem, we would need all t events $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$. However, storing these events requires huge space and solving the optimization problem from scratch is time consuming. Therefore, the online algorithm we propose here aims to finding the optimal dictionary $\mathbf{D}_t$ given $\mathbf{D}_{t-1}$ and $\mathbf{X}_t$. Specifically, we use projected first order stochastic gradient descent, consisting of the following update [88]:

$$\mathbf{D}_t = \Pi_{\mathcal{C}} \left[\mathbf{D}_{t-1} - \frac{\eta}{t} \nabla_{\mathbf{D}} l(\mathbf{X}_t, \mathbf{D}_{t-1}) \right] \tag{5.9}$$

where $l(\mathbf{X}_t, \mathbf{D}_{t-1}) = \frac{1}{2} \sum_{j=1,\ldots,n_t} ||\mathbf{X}_t^j - \mathbf{D}_{t-1} \boldsymbol{\alpha}_t^j||_2^2$, η is the learning rate, $\Pi_{\mathcal{C}}$ is the orthogonal projection onto $\mathcal{C}$.

5.2.4 Unusual Event Detection

As briefly mentioned in previous section, given a newly observed event $\mathbf{X}'$ and the current dictionary $\mathbf{D}^*$, the proposed algorithm learns the corresponding optimal reconstruction weight vector $\boldsymbol{\alpha}'$. $\mathbf{X}'$ is detected as an unusual event if the following criterion is satisfied

$$J(\mathbf{X}', \boldsymbol{\alpha}', \mathbf{D}^*) > \hat{\epsilon} \tag{5.10}$$

where $\hat{\epsilon}$ is a user defined threshold that controls the sensitivity of the algorithm to unusual events. Combining everything together, Algorithm 3 presents our unusual event detection method.

Algorithm 3 Unusual event detection using sparse coding

```
Input: video data, learning rate η, threshold ε̂
Learn initial dictionary using first N frames in video
repeat
  Use sliding window to obtain event X_t
  Learn optimal reconstruction vectors α_t for event X_t by solving Eq. (5.4) with D = D_{t-1}
  if J(X_t, α_t, D_{t-1}) > ε̂ then
    Fire alarm for event X_t
  end if
  Update dictionary D with Eq. (5.9)
until reach the end of video
```

5.3 Experiments

In this section, we show the empirical performance of the proposed unusual event detection algorithm, both qualitatively and quantitatively.

5.3.1 Subway Surveillance Video

The first 2 data sets are video sequences taken from surveillance camera at a subway station, with one camera monitoring the exit and the other monitoring the entrance. In both videos, there are

roughly 10 people walking around in a typical frame, with a frame size of 512×384. The videos are provided by courtesy of Adam et al. [1] and we compare quantitatively the detection results of our approach against the method in [65].

Subway Exit

The subway exit surveillance video is 43 minutes long with 64901 frames in total. To ensure a fair qualitative comparison, we follow the same definition of unusual events used in [65] for the same data set, though it should be noted that the definition of unusual events is rather subjective. Specifically, 3 types of unusual events are defined in the subway exit video: (a) walking in the wrong direction *(WD)*; (b) loitering near the exit *(LT)* and (c) misc, including suddenly stop and look around, janitor cleaning the wall, someone gets off the train and gets on again very soon. Totally, 19 unusual events are defined as ground truth.

We use a sliding window of size 80×80 pixels along x and y axes, and 40 frames along t axis in our approach. The fist 5 minutes of the video, same as in [65], is used to build initial dictionary. Before providing the unusual event detection results, we first show the dictionary learned using our approach in Figure 5.4. Specifically, Figure 5.4 visualizes randomly selected

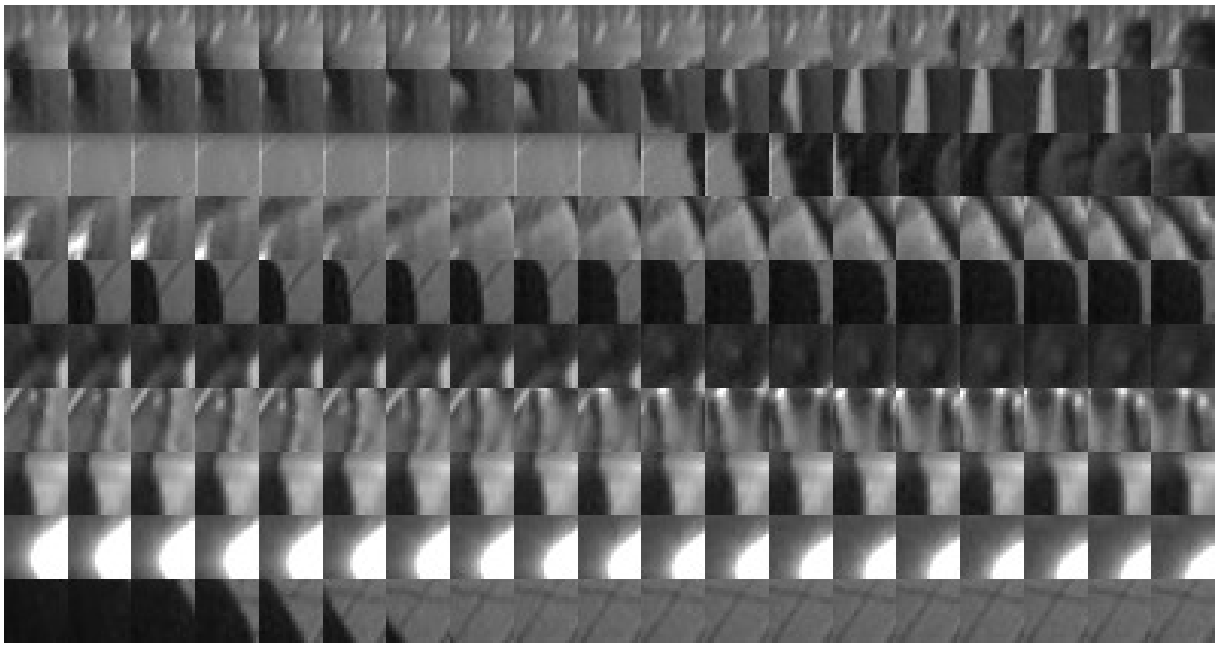

Figure 5.4: Dictionary learned using our approach for subway exit surveillance video. Each row in the figure corresponds to a basis in the dictionary. Typical activities in this dictionary include: walking to the left or right, walking towards the camera, train leaving station, etc.

10 bases in the learned dictionary (the size of the learned dictionary is 100). We observe that the learned bases of the dictionary reflects our intuition about what common and usual events are in this video: people walking towards the camera (exiting the subway), walking to the left or right, train leaving station, etc.

Table 5.1: Comparison of unusual event detection rate and false alarm rate on subway exit surveillance data: GT stands for ground truth annotation; ST-MRF refers to the method proposed in [65].

	WD	LT	MISC	Total	FA
GT	9	3	7	19	0
ST-MRF [65]	9	3	7	19	3
Ours	9	3	7	19	2

Table 5.1 provides quantitative results on unusual event detection accuracy and false alarm rate. We follow the same annotation used in [65], where a frame range is defined for each unusual event. For evaluation, once the algorithm detects at least one frame in the annotated range, the detection is counted as correct. On the other hand, false alarm is also measured in the same way: at least one frame is fired outside the annotated range, then it is counted as false alarm[2]. Figure 5.5 shows the detection results on the subway exit data, including the correct detections, and false alarms. Our method can detect an unusual event even within a crowded scene with occlusions (e.g., Figure 5.5(d)). Also, we can see that our method captures the unusual event caused by fine scale irregular motion (e.g., Figure 5.5(k)), or abnormal event resulted by irregular temporal ordering of activities (e.g., Figure 5.5(j)). We also illustrate two false alarms detected by our algorithm (Figure 5.5(o) & (p)). Curiously, looking closer into the video, these two events are indeed "unusual": Figure 5.5(o) is due to the first appearance of a child, and Figure 5.5(p) is due to the action of a man stopping near the exit and looking back. They are missed in ground truth annotations, hence labeled as FA in evaluation.

Time Complexity: We implement our algorithm using MATLAB 7.0 on a 2.60GHZ Intel CoreTM2 Duo PC with 2.0GB main memory. Learning initial dictionary took about 20 minutes. For each sliding window, learning reconstruction vectors took 0.2 seconds on average. In each frame, there are roughly 10 sliding windows being tested for unusual events. Consequently, unusual event detection in each frame was done in about 2 seconds.

Subway Entrance

The subway entrance video is 1 hour 36 minutes long with 144249 frames in total. 66 unusual events are defined, covering 5 different types: (a) walking in the wrong direction *(WD)*; (b) no payment *(NP)*; (c) loitering *(LT)*; (d) irregular interactions between people *(II)* and (e) misc, including sudden stop, running fast.

We use the same sliding window as in subway exit video, and the fist 15 minutes for training as in [65]. Figure 5.6 shows the dictionary learned by our approach, where we randomly select 12 bases out of 200 in the dictionary. This dictionary shows activities such as people walking to the left or right, walking away from the camera, which are usual events in this video. Quantitative comparison results with [65] are shown in Table 5.2, where our approach achieves higher detection rate and fewer false alarms. Moreover, as reported in [65], the approach in [1] fails to detect abnormal activities with irregular temporal orderings, such as Figure 5.5(j), people getting

[2]There are other evaluation metrics which could also be reasonable. We use this evaluation metric to be able to compare with [65].

off the train and getting back quickly. Also, the method in [1] results in an order magnitude more false alarms than [65]. Moreover, the clustering-based method [150] cannot detect events happening at a fine local scale, such as Figure 5.7(e) & (f). Therefore, while achieving slightly better qualitative performance than [65], our method also clearly outperforms the methods in [1] and [150] by a large margin.

Figure 5.7 displays unusual events detected using our approach. Our method not only detects abnormalities in a fine scale (e.g., Figure 5.7(e) & (f)), but also unusual events caused by irregular interactions between people (e.g., Figure 5.7(j)). Moreover, we can see that our method could correctly detect abnormal activities where both usual and unusual events occur in the same frame (e.g., Figure 5.7(g)).

Analysis Experiment: Online Update of the Learned Dictionary

In our approach, the learned dictionary is updated after observing each new event using projected stochastic gradient descent. In this section, we compare the results of our algorithm with the method using initially learned dictionary throughout the entire video sequence. Specifically, in the subway exit surveillance data, the second method learns an initial dictionary using the first 5 minutes of video and keep this dictionary fixed in the entire detection process. Similarly, in the subway entrance video data, the second method employs the fixed dictionary learned from first 15 minutes of video. Table 5.3 compares the detection accuracy and false alarms of the two methods. The method using fixed dictionary generally gives more false alarms than our approach. This result underscores our contribution in developing an online learning framework to update the bases dictionary. Without the online updates, the Fixed Dictionary method shows the inability for adapting to the changing contents of the video, resulting in a much greater error rate.

5.3.2 Unusual Event Detection in Youtube Videos

The above experiment has demonstrated our model's superiority in unusual event detection in surveillance videos, where the camera is fixed and the environment is relatively controlled. But our framework is a general approach that makes no assumptions of the cameras, the types of environment, or the contents of the video. In this section, we apply our method to a number of videos "in the wild", highlighting its application to a wide range of data. We downloaded a number of videos from YouTube. As Figure 5.8 shows, these videos have very different camera motion (rotation, zoom in/out, fast tracking, slow motion, etc.), contains different categories of targets (human, vehicles, animals, etc.) and covers a wide variety of activities and environmental conditions (indoor, outdoor).

For each of the 8 Youtube videos, we use approximately the first 1/5 of video data to learn an initial dictionary, and display detected unusual events in Figure 5.8. With no model assumptions of what is unusual, no need for templates, no supervision or training, our method could correctly detect abnormal activities in these real world low-quality videos.

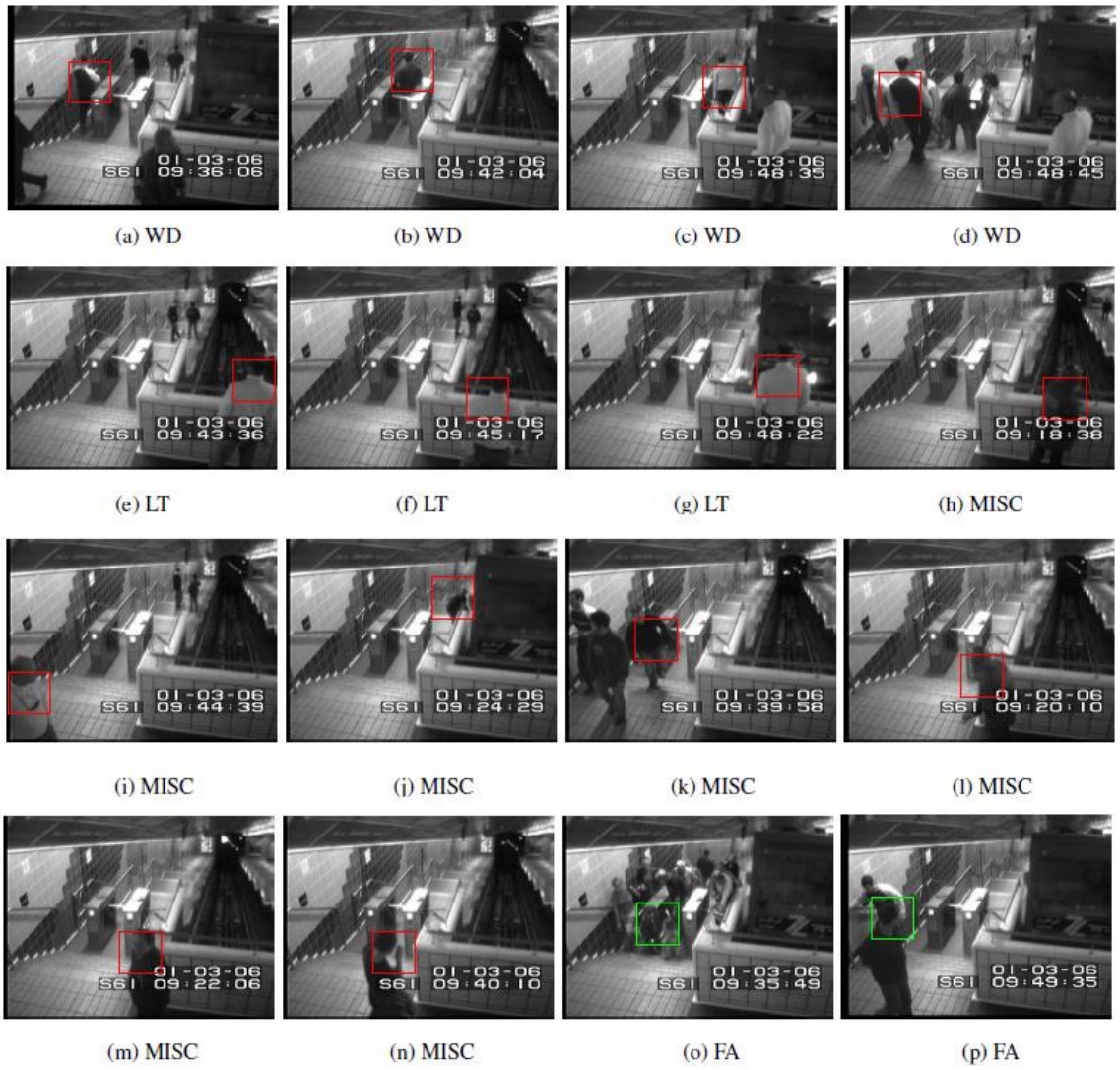

Figure 5.5: Unusual event detection in the subway exit surveillance video. WD: wrong direction; LT: loitering; MISC: misc; FA: false alarm. The rectangle on the figure marks the sliding window that results in the detection of unusual events. False alarms are marked using green sub-window.

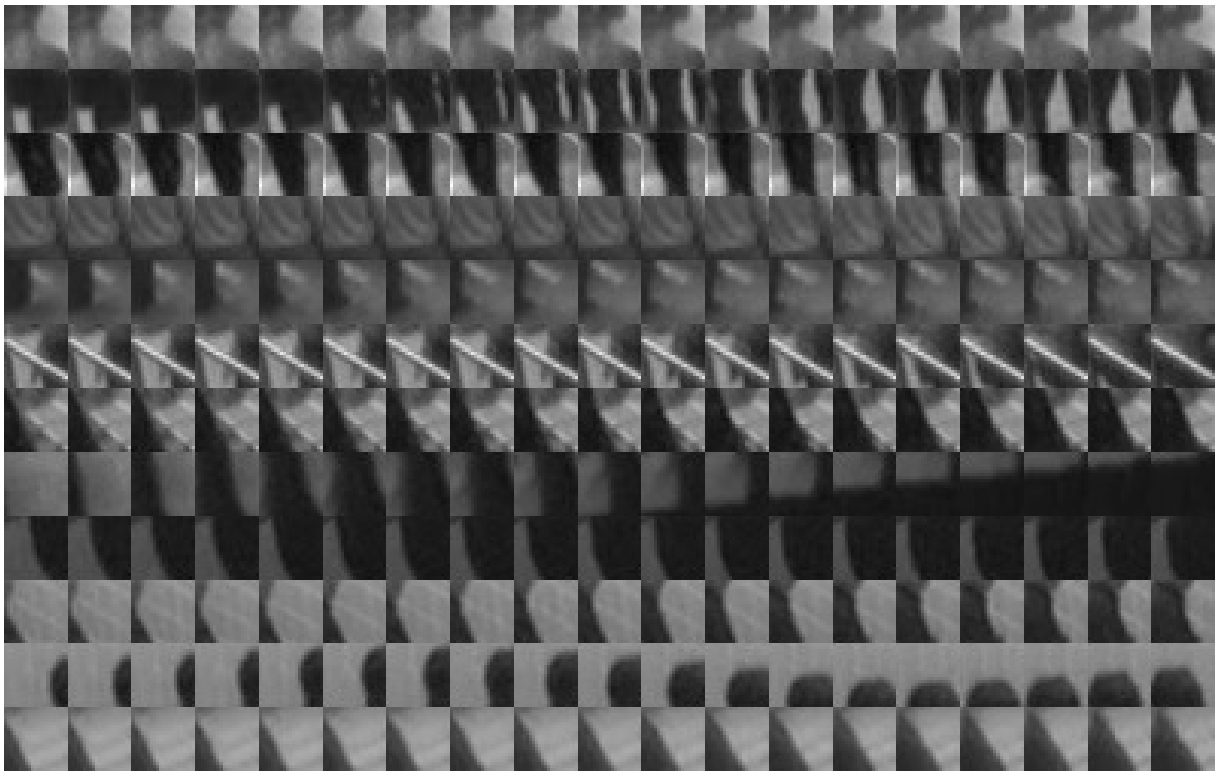

Figure 5.6: Dictionary learned using our approach for subway entrance surveillance data. Each row in the figure corresponds to a basis in the dictionary. Typical activities in this dictionary include: walking to the left or right, walking away from the camera, etc.

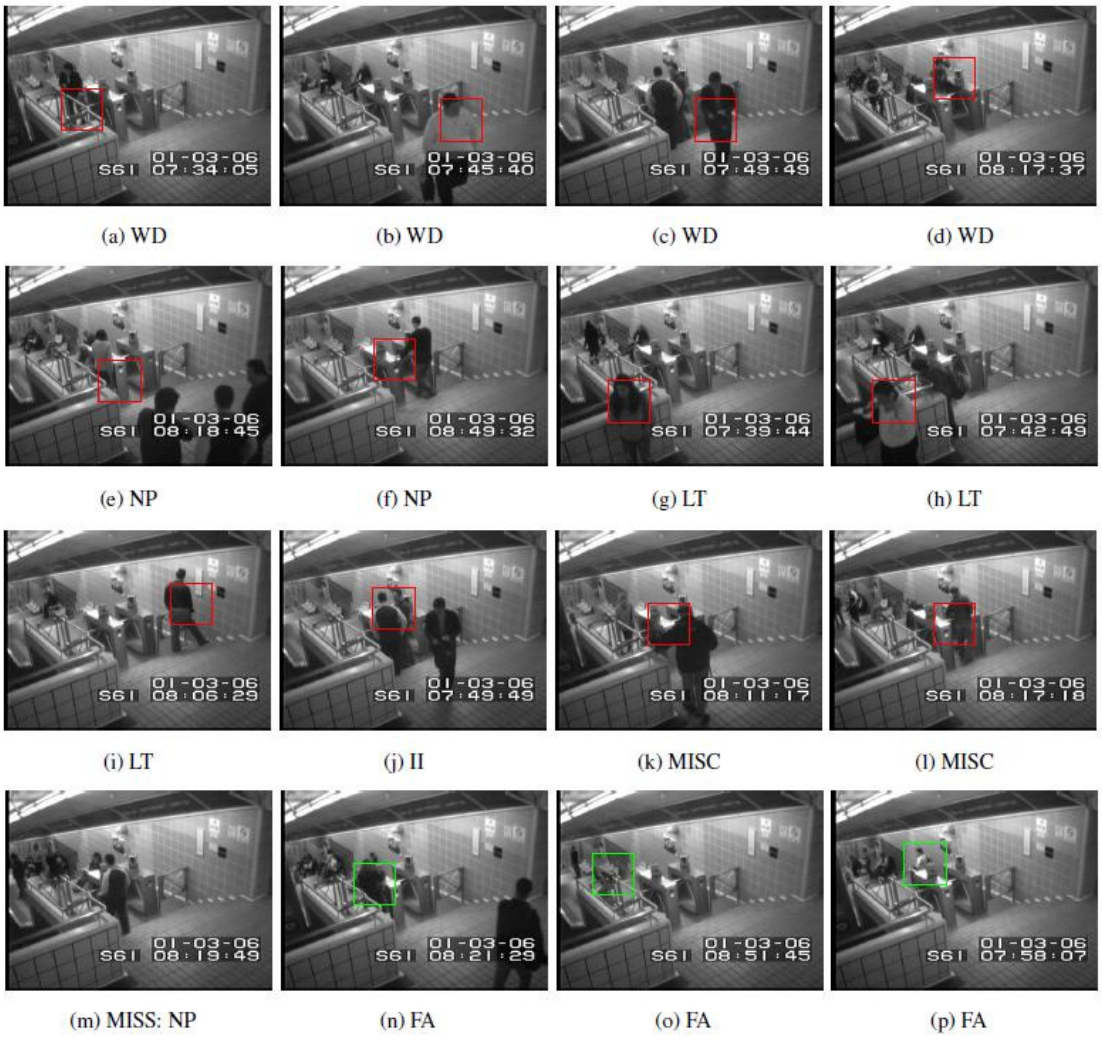

Figure 5.7: Unusual event detection in the subway entrance surveillance video. WD: wrong direction; NP: no payment; LT: loitering; II: irregular interactions; MISC: misc; MISS: missed unusual event; FA: false alarm.

Table 5.2: Comparison of unusual event detection rate and false alarm rate on subway entrance surveillance data.

	WD	NP	LT	II	MISC	Total	FA
GT	26	13	14	4	9	66	0
ST-MRF [65]	24	8	13	4	8	57	6
Ours	25	9	14	4	8	60	5

Table 5.3: Comparison of unusual event detection rate and false alarm rate: online updating dictionary vs. fixed dictionary. The number before '/' is for subway exit surveillance data, while the number after '/' is for entrance surveillance data.

	Correct Detection	False Alarm
Fixed D	17/54	8/12
Ours	19/60	2/5

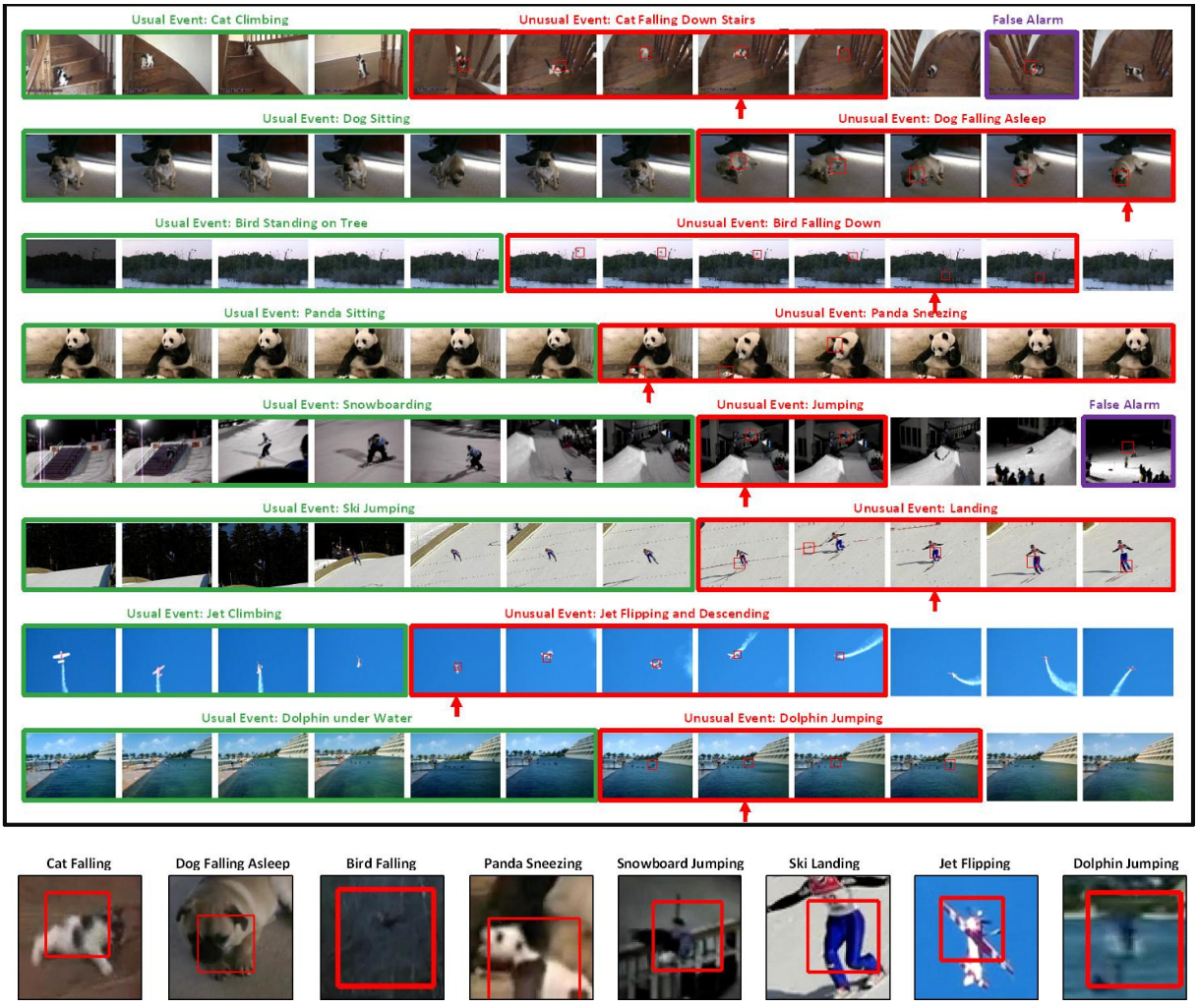

Figure 5.8: Unusual event detection results on 8 Youtube Videos. Frames of usual events, detected unusual events and false alarms are shown in the first 8 rows. For frames involving unusual events, red boxes on video frames represent patches that trigger the alarm. The bottom row provides a zoom-in view of those patches, taken from one frame (pointed by red arrows) per video.

5.4 Summary

We propose an unsupervised algorithm to automatically detect unusual events from a video sequence. A query video segment is projected onto a set of sparse coding bases learned by the algorithm, to obtain the reconstruction vectors. Typicality is then computed based on these reconstruction vectors. Moreover, the sparse coding bases are updated dynamically in an online fashion, to capture possible concept drift in video contents. Experimental results on two real world surveillance videos and several Youtube videos demonstrate the effectiveness of the proposed algorithm.

Chapter 6

Quasi Real-Time Summarization for Consumer Videos

In this chapter, we attempt to develop a method that offers the following function and alike: *"I only have 1 minute for this hour-long video, tell me where/what to watch"*. That is, it automatically compiles the most salient and informative portion of the video for users, by automatically scanning through video stream, in an online fashion, to remove repetitive and uninteresting contents. Our method differs from some previous attempts to video summarization that eliminate completely the time axis, and show a synopsis of the video by collecting a few key frames which are selected either arbitrarily, or according to some importance criteria [44, 78, 141]. Such key frame representation loses the dynamic aspect of video and is uninteresting to watch. More importantly, taking merely frames as the unit of content in a video omits much important information such as suspicious behaviors to be recognized automatically by a machine, and therefore compromises the quality of the summary. On the other hand, the summary generated by our proposed method is a short video itself, revealing the essence of the original video, just like a "trailer". In this chapter, we refer to those unstructured and unedited videos as *consumer videos*, in contrast to movies, news or sports videos which are often edited by human or having special structure (such as shot, scene, etc.).

6.1 Introduction

We propose *onLIne VidEo highLIGHTing (LiveLight)*, a principled way of online generation of a short video summarizing the most important and interesting contents of a potentially very long video. Specifically, *LiveLight* scans through the video stream, divided into a collection of video segments temporally. After processing the first few segments, it starts to build its own dictionary, which will be kept updated and refined later. Given a new video segment, *LiveLight* attempts to employ its current version of dictionary to sparsely reconstruct this previously unseen segment, using *group sparse coding* [8]. A small reconstruction error of the new video segment reflects that its content is already well represented in the current dictionary, further suggesting video segments containing similar contents have been observed in early part of the video. Hence, this segment is excluded from the summary, and the algorithm moves on to next segment. On the

other hand, if the new video segment cannot be sparsely reconstructed, i.e., a high reconstruction error is suffered, indicating unseen contents from previous video data, our method incorporates this video segment into the summary, and updates the dictionary according to the newly included video data. This process continues until the end of the video is reached. In summary, our method sequentially scans the video stream once, learns a dictionary to summarize contents seen in the video and updates it after encountering video data that could not be explained using current dictionary. A summary video is then constructed as a combination of two groups of video segments: (1) the first few segments used to learn initial dictionary, capturing background and early contents of the video; (2) video segments causing dictionary update, suggesting unseen and interesting contents. Moreover, as the entire process is carried out online, *LiveLight* could handle hours or even endless video data, ubiquitous in consumer videos.

6.2 Online Video Highlighting

Given an unedited and unstructured consumer video, *online video highlighting* starts with temporal segmentation, breaking original video into segments. Such temporal segmentation should ensure minimum variation, and consistency of objects, view and dynamics within each segment. Unlike structured videos, where shot boundary detection could be employed for temporal segmentation, most consumer videos do not even have such shot boundary, but instead with continuous camera movement. Therefore, we choose to evenly divide the original video into segments, each with a constant length of 50 frames (roughly $1 \sim 2$ seconds). Such short temporal length ensures the consistency within each segment. These video segments are the base units in *LiveLight*, in the sense that a few selected ones will compose the final summary video. A key component in *LiveLight* is the *dictionary*, which summarizes the contents of seen video. Specifically, a dictionary is initially learned using video segments at the beginning of the input video, with *group sparse coding*. After dictionary initialization, *LiveLight* scans through the rest of the video segments following temporal order, and attempts to reconstruct each video segment using the learned dictionary. Those video segments with reconstruction error higher than certain threshold are considered to contain interesting contents unprecedented in previous video, and are included into the summary video. Moreover, the dictionary is updated accordingly to incorporate the newly observed video contents, such that similar video segments seen later will suffer much smaller reconstruction error. On the other hand, those video segments that could be well reconstructed using the current dictionary are excluded from the summary, as small reconstruction error suggests its content is already well represented in the current dictionary, further indicating video segments containing similar contents have been observed in early part of the video. Hence, the dictionary represents the knowledge about previously seen video contents, and is updated in an online fashion to incorporate newly observed contents. Algorithm 4 provides the work flow of *LiveLight*, where $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ is used to learn initial dictionary with $m \ll K$, and ϵ_0 is a pre-set threshold parameter controlling length of the summary video.

Algorithm 4 Online Video Highlighting (LiveLight)

input Video $\mathcal{X}$ composed of temporal segments $\{\mathbf{X}_1, \ldots, \mathbf{X}_K\}$
output Short video $\mathcal{Z}$ summarizing most important and interesting contents of $\mathcal{X}$

1: Learn initial dictionary $\mathbf{D}$ using $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ via group sparse coding and initialize $\mathcal{Z} = \mathcal{X}_0$
2: **for all** Video segments $\mathbf{X}_k \in \{\mathbf{X}_{m+1}, \ldots, \mathbf{X}_K\}$ **do**
3: Reconstruct video segment $\mathbf{X}_k$ using current dictionary $\mathbf{D}$ and compute reconstruction error ϵ_k
4: **if** ϵ_k is larger than pre-set threshold ϵ_0 **then**
5: Update dictionary $\mathbf{D}$ using $\mathbf{X}_k$ and incorporate $\mathbf{X}_k$ into summary video $\mathcal{Z} = \mathcal{Z} \cup \mathbf{X}_k$
6: **end if**
7: **end for**

6.2.1 Video Segment Reconstruction

The basic idea for our approach is to represent the knowledge of previously observed video segments using the learned dictionary $\mathbf{D}$, whose columns (a.k.a. atoms) are bases for reconstructing future video segments. Given learned dictionary $\mathbf{D}$ (details of learning initial dictionary will be provided later in this section), *LiveLight* attempts to sparsely reconstruct query video segment using its atoms. Specifically, sparse reconstruction indicates both small reconstruction error and small footprint on the dictionary, i.e., using as few atoms from the dictionary as possible. Consequently, video summarization is formulated as a sparse coding problem, seeking linear decomposition of data using a few elements from a dictionary learned in online fashion.

We adopt the same feature representation for video data as used in Chapter 5. Specifically, we utilize the spatio-temporal interest points detected using the method in [40], and describe each detected interest point with histogram of gradient (HoG) and histogram of optical flow (HoF). The feature representation for each detected interest point is then obtained by concatenating the HoG feature vector and HoF feature vector. Finally, each video segment is represented as a collection of feature vectors, corresponding to detected interest points, i.e., $\mathbf{X}_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_k}\}$, where n_k is the number of interest points detected in video segment $\mathbf{X}_k$.

Different from conventional settings of sparse coding, where input signal is a vector, the input signal in our problem is a video segment, represented as a group of vectors $\mathbf{X}_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_k}\}$. Therefore, our goal is to effectively encode groups of instances in terms of a set of dictionary atoms $\mathbf{D} = \{\mathbf{d}_j\}_{j=1}^{|\mathbf{D}|}$, where $|\mathbf{D}|$ is the size of the dictionary, i.e., number of atoms in $\mathbf{D}$. Specifically, given learned dictionary $\mathbf{D}$, *LiveLight* seeks sparse reconstruction of the query segment $\mathbf{X}$, as follows

$$\min_{\mathbf{A}} \frac{1}{2} \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x}_i \in \mathbf{X}} \left\| \mathbf{x}_i - \sum_{j=1}^{|\mathbf{D}|} \boldsymbol{\alpha}_j^i \mathbf{d}_j \right\|_2^2 + \lambda \sum_{j=1}^{|\mathbf{D}|} \|\boldsymbol{\alpha}_j\|_2 \tag{6.1}$$

where $\mathbf{A} = \{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathbf{X}|}\}$, $\boldsymbol{\alpha}^i \in \mathbb{R}^{|\mathbf{D}|}$ is the reconstruction vector for interest point $\mathbf{x}_i \in \mathbf{X}$, and $|\mathbf{X}|$ is the number of interest points detected within video segment $\mathbf{X}$. The first term in (6.1) is reconstruction cost. If video segments similar to $\mathbf{X}$ have been observed before, this term should be small, due to the assumption that the learned dictionary represents knowledge

in the previously seen video data. The second term is the group sparsity regularization. Since dictionary $\mathbf{D}$ is learned to sparsely reconstruct previously seen video segments, if $\mathbf{X}$ contains no interesting or unseen contents, it should also be sparsely reconstructible using few atoms in $\mathbf{D}$. On the other hand, if contents in $\mathbf{X}$ have never been observed in previous video segments, although it is possible that a fairly small reconstruction cost could be achieved, we would expect using a large amount of video fragments for this reconstruction, resulting in dense reconstruction weight vectors. Moreover, the special mixed ℓ_1/ℓ_2 norm of $\mathbf{A}$ used in the second term regularizes the number of dictionary atoms used to reconstruct the entire video segment $\mathbf{X}$. This is more preferable over conventional ℓ_1 regularization, as a simple ℓ_1 regularizer only ensures sparse weight vector for each interest point $\mathbf{x}_i \in \mathbf{X}$, but it is highly possible that different interest points will have very different footprint on the dictionary, i.e., using very different atoms for sparse reconstruction. Consequently, reconstruction for the video segment $\mathbf{X}$ could still involve large number of atoms in $\mathbf{D}$. On the other hand, the ℓ_1/ℓ_2 regularizer ensures a small footprint of the entire video segment $\mathbf{X}$, as all interest points within segment $\mathbf{X}$ are regularized to use the same group of atoms for reconstruction. Moreover, the tradeoff between accurate reconstruction and compact encoding is controlled by regularization parameter λ. Finally, we denote the value of (6.1) with optimal reconstruction matrix $\mathbf{A}$ as ϵ, which is used in Algorithm 4 to decide if segment $\mathbf{X}$ should be incorporated into the summary video.

Consequently, *LiveLight* encapsulates the following intuitions for what we would think of a video summary. Given a dictionary optimized to sparsely reconstruct previously seen video contents, a new segment exhibiting similar contents seen in previous video data should be reconstructible from a small number of such atoms. On the other hand, a video segment unveiling contents never seen before is either not reconstructible from the dictionary of previous video segments with small error, or, even if it is reconstructible, it would necessarily build on a combination of a large number of atoms in the dictionary. Crucial to this technique, is the ability to learn a good dictionary of atoms representing contents seen in previous video segments, and being able to update the dictionary online to adapt to changing content of the video, which we discuss in detail later in this section.

Optimization

To find the optimal reconstruction vectors $\{\boldsymbol{\alpha}^i\}$ for interest points in $\mathbf{X}$, we need to solve problem (6.1). We employ *alternating direction method of multipliers (ADMM)* [20] to carry out such optimization, due to its efficiency. Specifically, *ADMM* consists of the following iterations:

$$\forall i: \ \boldsymbol{\alpha}^i_{k+1} = \left[\frac{\mathbf{D}^\top\mathbf{D}}{|\mathbf{X}|} + \rho\mathbf{I}\right]^{-1}\left[\frac{\mathbf{D}^\top\mathbf{x}_i}{|\mathbf{X}|} + \rho(\mathbf{z}^i_k - \mathbf{u}^i_k)\right] \tag{6.2}$$

$$\forall j: \ \mathbf{z}_{j,k+1} = \mathcal{S}_{\lambda/\rho}(\boldsymbol{\alpha}_{j,k+1} + \mathbf{u}_{j,k}) \tag{6.3}$$

$$\forall i: \ \mathbf{u}^i_{k+1} = \mathbf{u}^i_k + \boldsymbol{\alpha}^i_{k+1} - \mathbf{z}^i_{k+1} \tag{6.4}$$

and alternates among the above updates until convergence, where $\rho > 0$ is called the penalty parameter, $\mathbf{I} \in \mathbb{R}^{|\mathbf{D}|\times|\mathbf{D}|}$ is the identity matrix, $\mathcal{S}$ is the soft-thresholding operator defined as $\mathcal{S}_\kappa(a) = \max\{(1-\kappa/|a|), 0\}a$, $\mathbf{A} = \{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathbf{X}|}\}$, $\mathbf{Z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^{|\mathbf{X}|}\} = \{\mathbf{z}_1^\top, \ldots, \mathbf{z}_{|\mathbf{D}|}^\top\}^\top$, $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^{|\mathbf{X}|}\}$, index i runs through $\{1, \ldots, |\mathbf{X}|\}$, index j runs through $\{1, \ldots, |\mathbf{D}|\}$ and

k is the iteration counter. Both $\mathbf{Z}$ update (6.3) and $\mathbf{U}$ update (6.4) are trivial to compute. The $\mathbf{A}$ update in (6.2) can be accelerated via techniques such as warm start, caching factorization and fast matrix inversion as discussed in [20].

Learning Initial Dictionary

In this section, we discuss how to learn an initial dictionary, necessary to launch the *LiveLight* algorithm. Specifically, we would like a learning method that facilitates both induction of new dictionary atoms and removal of dictionary atoms with low predictive power. To achieve this goal, we again apply ℓ_1/ℓ_2 regularization, but this time to dictionary atoms. The idea for this regularization is that uninformative dictionary atoms will be regularized towards $\mathbf{0}$, effectively removing it from the dictionary. Given first few video segments $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, we formulate learning optimal initial dictionary as follows

$$\min_{\mathbf{D}, \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}} \frac{1}{m} \sum_{\mathbf{X}_k \in \mathcal{X}_0} J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \tag{6.5}$$

where $J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D})$ is the objective function in (6.1), and γ balances sparse reconstruction quality and dictionary size. Though non-convex to $\mathbf{D}$ and $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ jointly, (6.5) is convex w.r.t. $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ when $\mathbf{D}$ is fixed, and also convex w.r.t. $\mathbf{D}$ with fixed $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$. A natural solution is to alternate between these two variables, optimizing one while clamping the other. Specifically, with fixed dictionary $\mathbf{D}$, each $\mathbf{A}_k \in \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ can be optimized individually, using optimization method described in the previous section. On the other hand, with fixed $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$, optimizing dictionary $\mathbf{D}$ can be similarly solved via *ADMM*.

6.2.2 Online Dictionary Update

As *LiveLight* scans through the video, segments that cannot be sparsely reconstructed using current dictionary, indicating unseen and interesting contents, are incorporated into the summary video. However, all following occurrences of similar contents appearing in later video segments, should ideally be excluded. Consequently, it is crucial to update the dictionary such that those video segments already included in the summary video should no longer result in large reconstruction error. Assume the current version of summary is $\mathcal{Z}_t$, composed of t video segments $\{\mathbf{X}_k\}_{k=1}^t$, then the optimal dictionary is the solution of the following problem

$$\min_{\mathbf{D}} f(\mathbf{D}) = \min_{\mathbf{A}_1, \ldots, \mathbf{A}_t} \frac{1}{t} \sum_{\mathbf{X}_k \in \mathcal{Z}_t} J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \tag{6.6}$$

where we need to store feature representations $\{\mathbf{X}_k\}_{k=1}^t$ for all t segments in $\mathcal{Z}_t$. This might not cause problem for short videos, however, for hours of videos, especially surveillance videos running endlessly, storing these feature representations requires huge space. Moreover, solving the above optimization problem from scratch using alternating optimization for each dictionary update, is extremely time consuming, and would hinder the algorithm from applicable to real

world consumer videos. Therefore, *LiveLight* employs online learning for approximate and efficient dictionary update [89]. Specifically, instead of optimizing dictionary $\mathbf{D}$ and reconstruction coefficients $\{\mathbf{A}_1, \ldots, \mathbf{A}_t\}$ simultaneously, *LiveLight* aggregates the past information computed during the previous steps of the algorithm, namely the reconstruction coefficients $\{\hat{\mathbf{A}}_1, \ldots, \hat{\mathbf{A}}_t\}$ computed using previous versions of dictionary, and only optimizes $\mathbf{D}$ in problem (6.6). Therefore, the online dictionary update seeks to solve the following approximate optimization problem

$$\min_{\mathbf{D}} \hat{f}(\mathbf{D}) = \frac{1}{t} \sum_{\mathbf{X}_k \in \mathcal{Z}_t} J(\mathbf{X}_k, \hat{\mathbf{A}}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \tag{6.7}$$

It is easy to see that $\hat{f}(\mathbf{D})$ upper bounds $f(\mathbf{D})$ in problem (6.6). Moreover, theoretical analysis shown in the next section guarantees that $\hat{f}(\mathbf{D})$ and $f(\mathbf{D})$ converges to the same limit and consequently $\hat{f}(\mathbf{D})$ acts as a surrogate for $f(\mathbf{D})$. Moreover, it is easy to show that problem (6.7) could be equivalently reformulated as follows

$$\min_{\mathbf{D}} \frac{1}{2t} Tr(\mathbf{D}^\top \mathbf{D} \mathbf{P}_t) - \frac{1}{t} Tr(\mathbf{D}^\top \mathbf{Q}_t) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \tag{6.8}$$

where $Tr(\cdot)$ is matrix trace, $\mathbf{P}_t$ and $\mathbf{Q}_t$ are defined as

$$\mathbf{P}_t = \sum_{k=1}^{t} \sum_{\alpha^i \in \mathbf{A}_k} \boldsymbol{\alpha}^i {\boldsymbol{\alpha}^i}^\top, \ \mathbf{Q}_t = \sum_{k=1}^{t} \sum_{\alpha^i \in \mathbf{A}_k} \mathbf{x}_i {\boldsymbol{\alpha}^i}^\top \tag{6.9}$$

Therefore, there is no need to store $\{\hat{\mathbf{A}}_k\}_{k=1}^t$ or $\{\mathbf{X}_k\}_{k=1}^t$, as all necessary information is stored in $\mathbf{P}_t$ and $\mathbf{Q}_t$. Finally, problem (6.8) could be efficiently solved using *ADMM*.

6.2.3 Importance of Dictionary

Very recently, there have been attempts employing the idea of sparse reconstruction for video summarization [30, 44]. However, those approaches use the entire video itself as basis for reconstruction, instead of learning and updating a dictionary as concise summary of video contents. Using the entire video as reconstruction basis [30, 44], significantly increases the complexity of optimization and computational time, as shown later in the experiments, the approach in [30] takes nearly 10 times more CPU time than *LiveLight* on the same videos. Such heavy computational footprint hinders those approaches from being applied in temporally long consumer videos (actually, [30, 44] only used videos with at most several minutes in their empirical study). Moreover, [30, 44] have to see the entire video before starting to generate summary, eliminating the possibility of real-time summarization. On the other hand, the dictionary learned and updated in *LiveLight* concisely summarizes the contents of seen video, significantly reduces computational cost, and captures any concept drift in video streams.

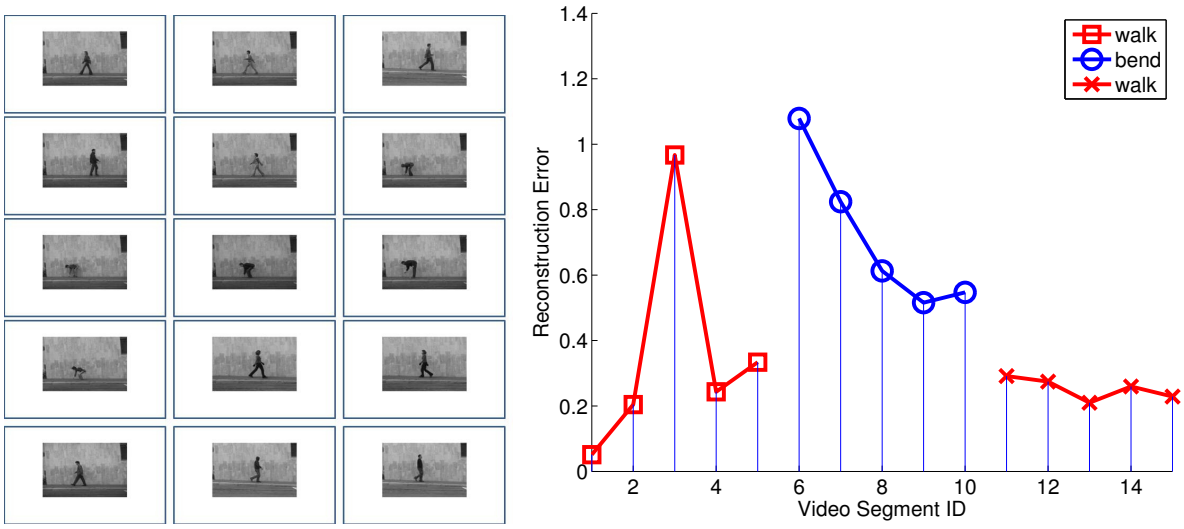

Figure 6.1: (Left) 15 video segments; (Right) reconstruction error for each video segment.

6.2.4 Sanity Check

We use synthetic video to perform sanity check on *LiveLight*. Specifically, we use two types of video sequences from *Weizmann* action recognition data [53], i.e., *walk* and *bend*. The synthetic video is constructed by combining 5 *walk* sequences, followed by 5 *bend* sequences, and 5 more *walk* sequences. Details of this synthetic video are shown in Figure 6.1. *LiveLight* learns initial dictionary using the first *walk* sequence, and carries out reconstruction and online dictionary update on the rest 14 video sequences. There are 2 clear peaks in Figure 6.1, corresponding to the third *walk* sequence, which is the first occurrence of walking from left to right (the first and second sequences are both walking from right to left), and the first *bend* sequence. Moreover, the reconstruction error for the fourth *walk* sequence, which also shows walking from left to right, is significantly smaller then the third *walk* sequence, indicating the dictionary has learned the contents of walking to the right, through online dictionary update. Finally, the last 5 *walk* sequences all result in small reconstruction errors, even after *LiveLight* has just observed 5 *bend* sequences, showing that the dictionary retains its knowledge about *walk*.

6.3 Theoretical Analysis

We first study the convergence property of online dictionary update. Specifically, we have the following theorem,

Theorem 2 *Denote the sequence of dictionaries learned in LiveLight as* $\{\mathbf{D}_t\}$*, where* $\mathbf{D}_1$ *is the initial dictionary. Then* $\hat{f}(\mathbf{D})$*, defined in (6.7), is the surrogate function of* $f(\mathbf{D})$*, defined in (6.6), satisfying*
(1) $f(\mathbf{D}) - \hat{f}(\mathbf{D})$ *converges to 0 almost surely;*
(2) $\mathbf{D}_t$ *obtained by optimizing* $\hat{f}$ *is asymptotically close to the set of stationary points of (6.6) with probability 1*

Theorem 2 guarantees that $\hat{f}(\mathbf{D})$ could be used as a proper surrogate for $f(\mathbf{D})$, such that we could optimize (6.7) to obtain the optimal dictionary efficiently, instead of solving the much more time-consuming optimization problem (6.6).

Next, we study the generalization ability of *LiveLight* on unseen video segments. Specifically, as *LiveLight* scans through the video sequence, the dictionary is learned and updated only using video segments seen so far. Consequently, the dictionary is optimized to sparsely reconstruct contents in seen video segments. It is crucial for *LiveLight* to also be able to sparsely reconstruct unseen video segments, composed of contents similar to video segments seen before. This property is called generalization ability in statistical machine learning terminology. Specifically,

Theorem 3 *Assume data points* $\mathbf{X}$ *(i.e., video segments) are generated from unknown probability distribution* $\mathcal{P}$*. Given* t *observations* $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$*, for any dictionary* $\mathbf{D}$*, and any fixed* $\delta > 0$*, with probability at least* $1 - \delta$

$$\mathbb{E}_{\mathbf{X}\sim\mathcal{P}} J^*(\mathbf{X}, \mathbf{D}) - \frac{1}{t}\sum_{k=1}^{t} J^*(\mathbf{X}_k, \mathbf{D}) \leq \epsilon(t, \delta) \tag{6.10}$$

where $J^*(\mathbf{X}, \mathbf{D}) = \min_{\mathbf{A}} J(\mathbf{X}, \mathbf{A}, \mathbf{D})$ *is the minimal reconstruction error for* $\mathbf{X}$ *using dictionary* $\mathbf{D}$*, as defined in (6.1), and* $\epsilon(t, \delta) = o(\ln t/\sqrt{t})$ *is a small constant that decreases as* t *increases.*

The above theorem is true for any dictionary $\mathbf{D}$, and obviously also true for the dictionary learned in *LiveLight*. Therefore, Theorem 3 guarantees that if dictionary $\mathbf{D}$ has small reconstruction error on previously seen video segments, it will also result in small reconstruction error for unseen video segments with similar contents. It should be noted that in *LiveLight*, the dictionary is learned in an online fashion, and a new dictionary is learned whenever a new video segment is incorporated into the summary video. Consequently, to correctly understand the implication of the above theorem, we should apply it to each dictionary learned through the online learning process. Specifically, assume the current version of summary is $\mathcal{Z}_t$, composed of t video segments $\{\mathbf{X}_k\}_{k=1}^t$, and the updated optimal dictionary is $\mathbf{D}_t$. Theorem 3 proves that if dictionary $\mathbf{D}_t$ has small reconstruction error on previously seen video segments (up until time point t), it will also result in small reconstruction error for unseen video segments with similar contents. Since the above logic is true for any dictionary $\mathbf{D}_t$ learned in the online process, it guarantees we could use the sparse re-constructibility to filter out events similar to previously seen video segments.

6.4 Experiments

We test the performance of *LiveLight* on more than 12 hours of consumer videos, including both YouTube videos and surveillance videos. The 20 videos in our data set span a wide variety of scenarios: indoor and outdoor, moving camera and still camera, with and without camera zoom in/out, with different categories of targets (human, vehicles, planes, animals etc.) and covers a wide variety of activities and environmental conditions. Details about the data set are provided in Table 6.1.

Table 6.1: Data set details. The first 15 videos are downloaded from YouTube, and the last 5 videos are from surveillance cameras. Video length (Time) is measured in minutes. *CamMo* stands for camera motion, and *Zoom* means camera zoom in/out.

Video	Time	Frames	CamMo	Zoom
CarRace	34.00	61133	Yes	No
FirefighterSave	21.53	38714	Yes	Yes
MonsterTruck	50.85	91430	Yes	No
StockCar	38.91	69963	Yes	No
SpeedBoat	64.03	115249	Yes	Yes
DogSwimming	20.08	36106	No	No
PetEvent	33.63	60472	Yes	Yes
HorseTraining	20.67	37171	Yes	No
ShowJumping	20.73	31087	Yes	Yes
Snorkeling	21.92	39463	Yes	No
DisneyParade	29.10	50694	Yes	No
ShamuShow	21.08	37845	Yes	No
BoatTour	28.36	51043	Yes	Yes
AirShow	12.04	21626	Yes	Yes
PolicePullOver	46.58	83761	Yes	No
SubwayExit	43.27	64902	No	No
SubwayEntrance	96.17	144249	No	No
Mall-1	55.44	83156	No	No
Mall-2	39.98	59969	No	No
Mall-3	60.35	90525	No	No

6.4.1 Experiment Design and Evaluation

We compare *LiveLight* with several other methods, including *evenly spaced segments* and *K-means clustering* [30] using the same features as our method. Specifically, after the temporal segmentation of the original video, evenly divided into segments each with a constant length of 50 frames, *evenly spaced segments* computes the number of segments N via dividing the length of ground truth summary video by the length of each video segment, i.e., 50 frames, then select N segments starting from the first segment with even space between any two segments. Those evenly spaced segments are then concatenated to compose the summary video for the *evenly spaced segments* method. For the *K-means clustering* method, we first group the collection of video segments from the original video into N clusters, using *K-means clustering*, where N is computed the same way as in the *evenly spaced segments* method. Then for each center of the N clusters, we find the segment from the original video that has minimum distance to that cluster center. Finally, the summary video for *K-means clustering* method is composed by concatenating the N video segments with minimum distance to the N cluster centers. We also compare with the *DSVS* algorithm proposed in [30], state-of-the-art method for unsupervised video summarization, using the 50-frame video segments as basic units in the algorithm. It is shown in [30] that *DSVS* already beats color-histogram based method [109] and motion-based

Table 6.2: T is the length (seconds) of summary video. LL: LiveLight; ES: evenly spaced segments; CL: K-Means Clustering; DSVS: sparse reconstruction using original video as basis [30].

	T	LL(%)	ES(%)	CL(%)	DSVS(%)
CarRace	60.7	**59.80**	40.53	44.98	58.04
FirefighterSave	59.4	**66.84**	38.22	52.53	55.83
MonsterTruck	61.6	**54.38**	42.05	49.35	46.36
StockCar	60.2	**61.30**	35.55	40.03	48.36
SpeedBoat	59.3	75.55	45.03	59.53	**77.07**
DogSwimming	62.2	**79.10**	42.93	54.34	64.57
PetEvent	61.9	**58.32**	36.51	54.12	43.93
HorseTraining	60.3	**66.17**	54.06	53.90	60.63
ShowJumping	60.6	**46.86**	39.93	37.13	44.22
Snorkeling	60.4	60.10	37.58	**62.09**	56.83
DisneyParade	58.5	**67.69**	39.49	62.56	67.17
ShamuShow	59.0	**61.36**	34.75	21.02	48.75
BoatTour	59.9	**59.28**	33.89	40.57	50.93
AirShow	36.8	**85.33**	49.46	74.46	72.29
PolicePullOver	61.7	**91.41**	30.63	60.45	76.75
SubwayExit	89.1	**91.87**	21.55	47.70	85.03
SubwayEntrance	92.1	**80.56**	42.56	43.65	73.85
Mall-1	89.7	**94.98**	37.46	57.08	85.40
Mall-2	88.4	**96.83**	43.67	61.99	87.72
Mall-3	92.0	**88.26**	38.70	58.59	81.99
Average	-	**72.30**	39.23	51.80	64.29

method [85]. As explained above, parameters for various algorithms are set such that the length of generated summary videos are the same as ground truth summary video. For *LiveLight*, we fix the number of atoms in dictionary to 200, though better performance is possible with fine tuning of parameters.

For each video in our data set, three judges selected segments from original video to compose their preferred version of summary video. The final ground truth is then constructed by pooling together those segments selected by at least two judges. Following [30], to quantitatively determine the overlap between algorithm generated summary and ground truth, both video segment content and time differences are considered. More precisely, we augment the ground truth summary video via potentially extending each segment by 2 seconds before and after it. Specifically, the ground truth summary video is composed by a series of video segments from the original video. For the i-th video segment in the ground truth summary video, we look at the video contents 2 seconds before it. If this 2 seconds of video contents show similar scene content and motion pattern as the i-th video segment, we add it into the augmented ground truth. We perform the same procedure for the video contents 2 seconds after the i-th video segment. Consequently, the final augmented ground truth summary video contains the original user generated ground truth summary video, and those 2 seconds of video contents which are showing similar scene

content and motion pattern as their corresponding ground truth video segment. Final accuracy is computed as the ratio of the number of overlapped frames between the augmented ground truth summary video and algorithm generated summary video, divided by the length of the original ground truth summary video.

6.4.2 Results

According to the quantitative comparison provided in Table 6.2, we have following observations: (1) *LiveLight* achieves highest accuracy on 18 out of 20 videos, and in most cases beats competing algorithms with a significant margin; (2) On the 5 surveillance videos, both *LiveLight* and *DSVS* outperform other two algorithms, showing the advantage of sparse reconstruction based methods on summarizing surveillance videos; (3) Averaged across 20 videos, *LiveLight* outperforms the state-of-the-art summarization method *DSVS* by 8%, revealing the advantage of *LiveLight*.

Besides quantitative measures, we also show the automatically generated summary ("trailer") for YouTube video *PolicePullOver* (more summary videos are provided in Figure 6.3, Figure 6.4, and Figure 6.5). As shown in Figure 6.2, the summary video captures the entire story line of this near hour long video, achieving more than 40 times compression in time without losing semantic understandability of the summary video. Moreover, the background in this video involves various cars passing in both directions, and it is interesting that *LiveLight* is not affected by this background motion.

6.4.3 Time Complexity

LiveLight is implemented using MATLAB 7.12 on a 3.40 GHZ Intel Core i7 PC with 16.0 GB main memory. Table 6.3 compares the processing time of various algorithms, with the following observations: (1) The last column under *LiveLight* shows the ratio between its computational time and video length. For all videos, this ratio is less than 2, and for 6 videos even less than 1. Thus, with MATLAB implementation on a conventional PC, *LiveLight* already achieves near real-time speed, further revealing its promise in real world video analysis applications; (2) *LiveLight* is nearly 10 times faster than *DSVS*, revealing the advantage of learning and updating dictionary in an online fashion, instead of using original video as basis for sparse reconstruction.

Table 6.3: Processing time of *LiveLight* and competing algorithms (all time shown is in minutes). T_{video} is the length of original video. T_1 is the time spent on generating feature representations in *LiveLight*, and T_2 is the combined time spent on learning initial dictionary, video segment reconstruction and online dictionary update. $T_{total} = T_1 + T_2$ is the total processing time of *LiveLight*, and $Ratio = T_{total}/T_{video}$ for all algorithms.

	Video	LiveLight				EvenlySpaced		Clustering		DSVS	
	T_{video}	T_1	T_2	T_{total}	Ratio	T_{total}	Ratio	T_{total}	Ratio	T_{total}	Ratio
CarRace	34.00	42.73	10.18	52.91	**1.56**	0.52	0.02	114.03	3.35	636.7	18.73
FirefighterSave	21.53	27.26	8.54	35.80	**1.66**	0.51	0.02	78.92	3.67	434.8	20.20
MonsterTruck	50.85	60.23	11.60	71.83	**1.41**	0.59	0.01	162.23	3.19	783.3	15.40
StockCar	38.91	51.95	10.08	62.03	**1.59**	0.50	0.01	150.59	3.87	672.2	17.27
SpeedBoat	64.03	39.65	9.31	48.96	**0.76**	0.54	0.01	102.56	1.60	460.2	7.19
DogSwimming	20.08	5.16	6.93	12.09	**0.60**	0.53	0.03	33.46	1.67	174.7	8.70
PetEvent	33.63	34.27	8.24	42.51	**1.26**	0.60	0.02	101.77	3.03	516.2	15.35
HorseTraining	20.67	9.82	7.64	17.46	**0.84**	0.56	0.03	34.84	1.69	271.8	13.15
ShowJumping	20.73	23.33	7.57	30.90	**1.49**	0.55	0.03	79.72	3.85	247.2	11.93
Snorkeling	21.92	28.30	9.13	37.43	**1.71**	0.54	0.03	88.19	4.02	536.6	24.48
DisneyParade	29.10	37.53	9.48	47.01	**1.62**	0.49	0.02	123.96	4.26	535.9	18.42
ShamuShow	21.08	17.91	7.83	25.74	**1.22**	0.55	0.03	51.46	2.44	399.0	18.93
BoatTour	28.36	25.56	9.13	34.69	**1.22**	0.50	0.02	80.48	2.84	379.8	13.39
AirShow	12.04	8.37	7.24	15.61	**1.30**	0.31	0.03	42.34	3.52	225.0	18.68
PolicePullOver	46.58	16.07	8.33	24.40	**0.52**	0.56	0.01	53.24	1.14	493.3	10.59
SubwayExit	43.27	14.46	6.67	21.13	**0.49**	0.75	0.02	148.8	3.44	558.6	12.91
SubwayEntrance	96.17	41.30	9.55	50.85	**0.53**	0.84	0.01	304.15	3.16	1299.1	13.51
Mall-1	55.44	51.02	9.93	60.95	**1.10**	0.82	0.02	139.23	2.51	646.4	11.66
Mall-2	39.98	33.49	8.95	42.44	**1.06**	0.80	0.02	94.75	2.37	436.5	10.92
Mall-3	60.35	58.02	10.31	68.33	**1.13**	0.78	0.01	137.20	2.27	698.4	11.57

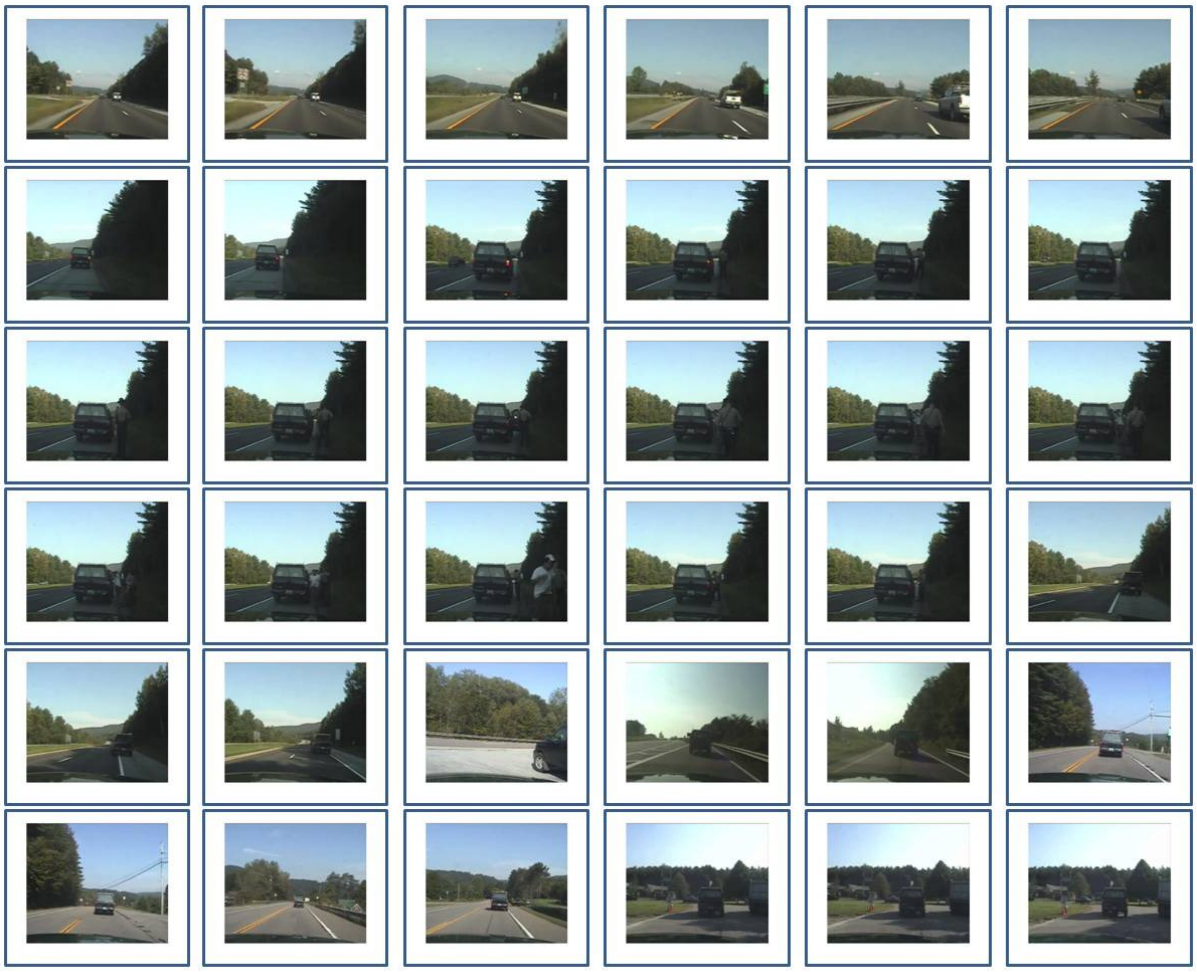

Figure 6.2: (Best viewed in color and zoom-in.) Some frames of the summary video generated by *LiveLight* for a YouTube video showing police pulling over a black SUV and making arrest (frames are organized from left to right, then top to bottom in temporal order). From the summary video, we could see the following storyline of the video: (1) Police car travels on the highway; (2) Police car pulls over black SUV; (3) Police officer talks to passenger in the SUV; (4) Two police officers walk up to the SUV, and open the passenger side door of the SUV; (5) Police officer makes arrest of a man in white shirt; (6) Police officer talks to passenger in the SUV again; (7) Both police car and black SUV pull into highway traffic; (8) Police car follows black SUV off the highway; (9) Both vehicles travel in local traffic; (10) Black SUV pulls into local community.

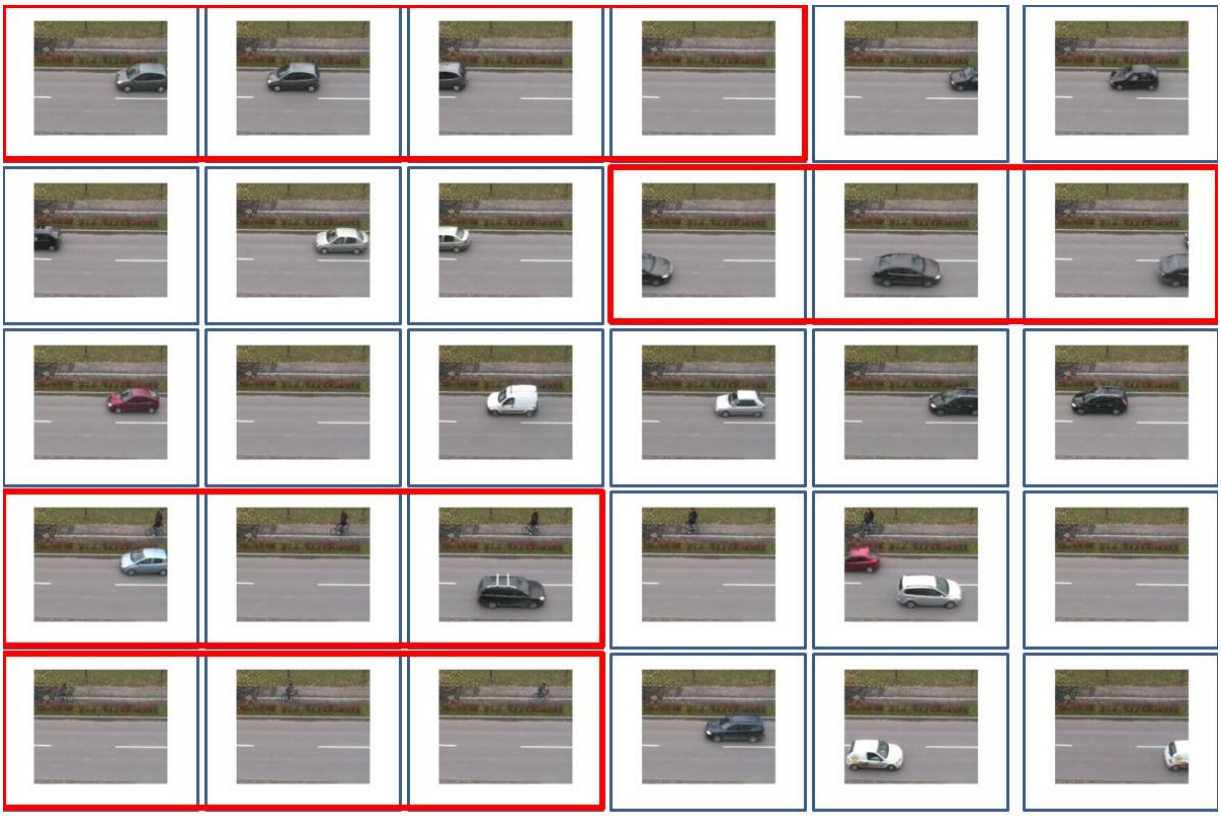

Figure 6.3: (Best viewed in color and zoom-in) Some frames of the traffic surveillance video and the video highlight generated by *LiveLight*. The video segments incorporated in the video highlight are shown in the red bounding boxes: (1) A car travels from right to left; (2) A car travels from left to right; (3) Two people push a bike from right to left; (4) A person rides a bike from left to right.

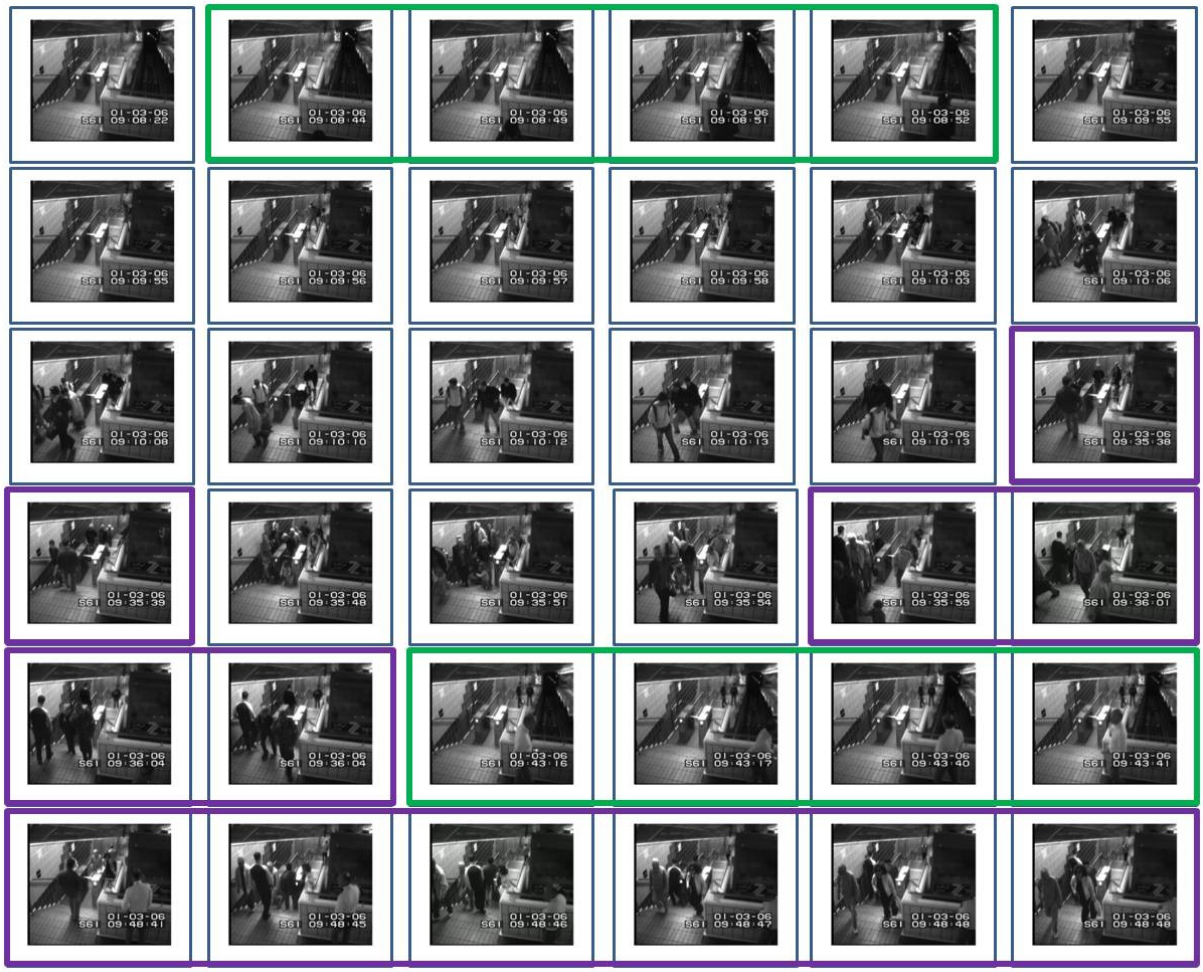

Figure 6.4: (Best viewed in color and zoom-in) Some frames of the video highlight for subway surveillance video. Besides showing people getting off the train and exiting the station, the video highlight also captures suspicious behaviors. Specifically, frames in purple bounding boxes show people walking in the wrong direction, i.e., getting into the station through exit, and frames in green bounding boxes show loitering near the exit.

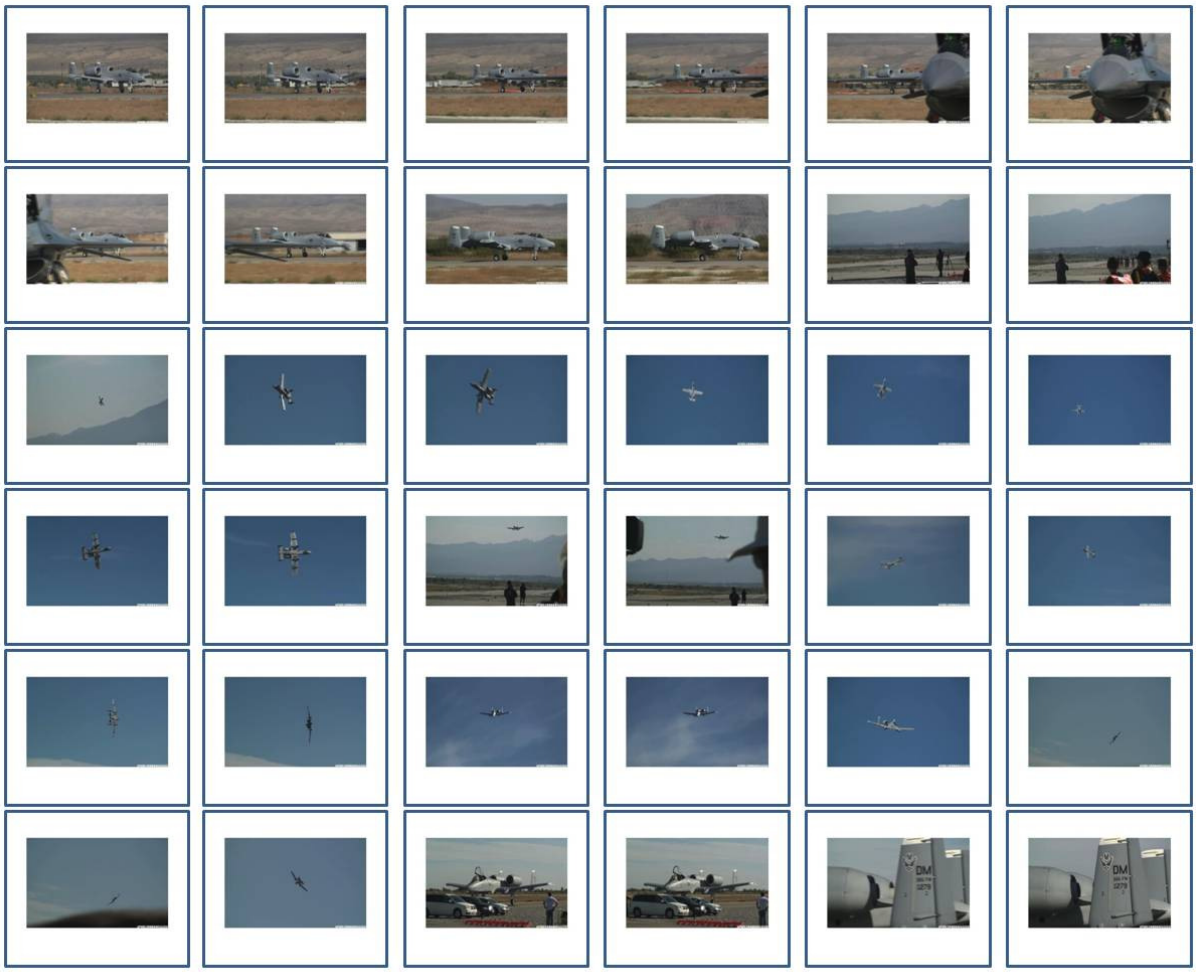

Figure 6.5: (Best viewed in color and zoom-in) Some frames of the video highlight for air show video. From the video highlight, we could see the following storyline of the video: (1) The plane starts taking off; (2) The plane passes another plane during taking off; (3) The plane takes off; (4) Other people watching the air show caught on camera; (5) The plane performs various stunts, including flying side-way, flying upside down, diving close to the ground, etc.; (6) The plane lands. It should be noted that at the end of the video highlight, it seems that *LiveLight* did not capture the process of landing. However, the reason for lacking such process is because the original video does not have this part at all.

6.5 Summary

In this chapter, we propose *LiveLight* to generate short video summarizing the most important and interesting contents, of a potentially very long video, and enable viewer to understand the video without watching the entire sequence. Theoretical analysis is provided, focusing on online dictionary update convergence, and generalization ability to unseen video segments. We summarize our main contributions as follows. (1) We propose a principled way of generating short video highlight of a potentially very long video, summarizing its most important and interesting contents while eliminating repetitive events, enabling viewer to understand the video without watching the entire sequence. (2) We propose an online dictionary update method, enabling our method to generate highlights on-the-fly. (3) We provide theoretical analysis of the proposed method, guaranteeing convergence of the online dictionary update and generalization ability to unseen video segments. (4) We demonstrate the effectiveness of *LiveLight* on real-world data, including both surveillance videos and YouTube videos, achieving near real-time speed on all tested videos.

The purely unsupervised formulation for video summarization in this chapter enables wide applicability of *LiveLight*. However, in videos recording certain events with clear definition of a storyline, such as a wedding video, human guidance on the video summarization process could be valuable, in order to avoid missing important piece of the storyline, to including unimportant segments. Therefore, in next chapter, we study video summarization in the supervised setting, where user generated ground truth annotation of summary video is provided for a subset of videos, and we propose an algorithm to automatically learn the preferred storyline and generate summary for videos of similar nature.

Chapter 7

Supervised Video Summarization: A Max Margin Approach

As discussed in previous chapter, video summarization, whose goal is to concisely present the storyline of the original video through removing significant portion of the original video that are deemed uninteresting or repetitive, has become not only a popular research topic in computer vision, but also a highly-anticipated tool for unlocking video data. However, despite the recently increasing interests on video summarization for consumer videos, almost all previous attempts have taken an unsupervised formulation, that is, only the video itself is used in deciding which portion should be kept in the summary. Albeit being generally applicable to any video stream, unsupervised video summarization has serious limitations. Specifically, most unsupervised video summarization algorithms aim at picking out unique and non-repetitive video segments to construct the summary video. However, uniqueness is not necessarily the optimal criterion for selecting video segments. For example, in a video captured by a hand-held device, if the user accidently drops the camera while recording, there will be a rather unique segment recorded during the process of camera being dropped. In most scenarios, such segments would be regarded as noise, and users would expect such accidental segments be deleted since they are not key components of the storyline in the original video. However, under the unsupervised video summarization framework, due to the uniqueness of such accidental segments, they would be deemed as highly interesting and very likely to be included into the final summary video. On the other hand, selecting most unique segments does not guarantee the constructed summary video could illustrate the storyline of the original video. For example, when summarizing wedding videos, users would expect a storyline consisting of events such as bride walking down the aisle, ring exchange, couple's first dance and cutting the wedding cake, included in the summary video. Unsupervised video summarization could possibly miss one or a few of such key events, and hence generating an incomplete summary video. Finally, given the same video, different users might have various preferences of which events should be incorporated into the summary video, unfortunately, unsupervised video summarization cannot utilize such information for personalized video summarization.

Clearly, some gap might exist between the automated unsupervised characterization of the important video segments and the human perception about the importance of a segment, and purely unsupervised video summarization could potentially miss some of these key components,

or incorporate unimportant events. Therefore, it is beneficial to have users generate their preferred summaries, and use such manually composed ideal summary video as supervised information for summarizing future videos characterizing similar events. Specifically, within a class of videos of similar nature, such as a collection of wedding videos, users could provide the desired summaries for a subset of videos. Based on such supervised information, the summaries for other videos in the same class are automatically generated. Such technique not only captures the user perception through the process of supervised learning, but also provides an algorithmic framework for generation of personalized video summary.

7.1 Introduction

In this chapter, we propose *GUIded viDEo SUMmarization (GuideSum)* for supervised video summarization (problem formulation shown in Figure 7.1), where users are asked to generate summaries for a subset of a class of videos of similar nature. Given such manually generated summaries, *GuideSum* aims to learn the preferred storyline within the given class of videos of similar nature. Then, *GuideSum* will automatically generate summaries for the rest of videos in the class, capturing the similar storyline in those manually summarized videos. One clear challenge in supervised video summarization lies in the fact that the number of unique events in the storyline is unknown, since the only supervised information is the summary video itself, without semantic labeling on each segment included into the summary. Moreover, while some videos present the entire sequence of certain events, others might only have partial events. Take the wedding videos as an example, some videos could show the entire process of couple cutting the wedding cake, others might only show the cake without actually cutting it. Therefore, the algorithm should understand not only entire sequence, but also partial sequences. Weighing the above challenges, our proposed *GuideSum* algorithm is formulated as a max margin classification problem, where those segments incorporated into the summary video are treated as positive training data, and segments excluded from the summary are considered as negative examples. In particular, *GuideSum* employs an ℓ_1/ℓ_2 regularization on the set of weight vectors to automatically select the optimal number of unique events, and augments the set of positive training examples by including not only full sequence but also partial sequences of each unique event, enabling understanding of partial events. Mathematically, *GuideSum* is formulated as an ℓ_1/ℓ_2 regularized max margin optimization problem, with exponential number of constraints. We utilize the cutting plane algorithm to effectively handle the huge number of constraints and *ADMM* to efficiently solve the ℓ_1/ℓ_2 regularized optimization problem. Experimental results on a collection of surveillance and YouTube videos are provided to demonstrate the effectiveness of *GuideSum*.

Our main contributions in this chapter are as follows. (1) We propose a principled way of generating short summary video of a potentially very long video, leveraging information from not only the original video itself, but also user generated summaries for videos capturing events of similar nature (such as a set of wedding videos), filling the gap between the automated characterization of the important video segments and the human perception about the importance of a segment. (2) In our formulation, the supervised information required is only the user generated summary video, with no requirement on semantic labeling for each segment incorporated into

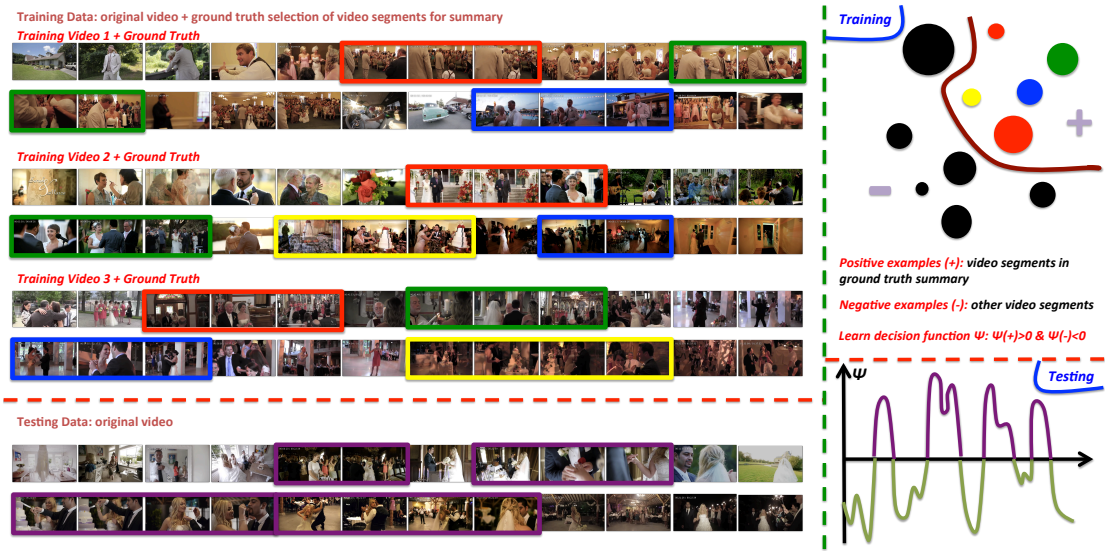

Figure 7.1: (Best viewed in color) Formulation of suervised video summarization. Within a class of videos of similar nature (such as a collection of wedding videos as shown in the figure), users provide the desired summaries for a subset of videos to be used as training data. Based on such supervised information, a max margin classifier Ψ is learned where those video segments incorporated into the user generated summary are treated as positive examples, while other video segments are utilized as negative examples. To automatically generate summary for unseen video, i.e., testing data, the learned max margin classifier is applied to the testing video, where any video segment with positive value for the decision function is incorporated into the summary video.

the summary video, making the process of collecting supervised information much simpler and rendering our proposed approach more practical. (3) We propose a max margin learning formulation for supervised video summarization, capable of automatically determining the optimal number of unique events in the summary video, and understanding not only full event but also partial events. (4) We demonstrate the effectiveness of *GuideSum* on real-world data, including both surveillance videos and YouTube videos, achieving a significant margin on summarization accuracy over state-of-the-art video summarization algorithms on all tested videos.

7.2 Supervised Video Summarization

For the vast majority of consumer generated videos recording events or life moments of special interests, there is usually an embedded storyline in the visually complicated video. For example, in wedding videos, the storyline could start with bride walking down the aisle, followed by the exchange of vows, ring ceremony, then couple's first dance, and cutting the wedding cake. The motivation for video summarization is to remove uninteresting events and retain only the most salient and important segments for the storyline, so that people could save significant amount of

time by watching only the summary video, which is much shorter than the original video, but at the mean time keeps all crucial segments that are key to the flow of the storyline. Clearly, unsupervised video summarization algorithms are prone to missing some of these key components, or incorporating unimportant events, due to the lack of knowledge about which events are key to the storyline. Consequently, serious gap exists between automated characterization of the important video segments and the human perception about the importance of a segment. Weighing the above issue with unsupervised video summarization, the motivation for our proposed max margin video summarization algorithm is to bridge the gap between machine perception of video segment importance and user preference on the storyline, with minimal requirement of supervised information.

Within a class of videos of similar nature, such as a collection of wedding videos, a naive way of incorporating user input as supervised information, is to ask users to identify each component of the storyline from the video sequence, and then use event recognition to separately identify each component of the storyline in the testing video. For example, a user might be asked to label out the video segment corresponding to ring ceremony from a wedding video. Though such supervised information could potentially boost the accuracy of video summarization, it also has serious limitations. To begin with, a crucial prerequisite for such formulation is the clear definition of key components in the storyline (for wedding videos: bride walking down the aisle, exchange of vows, ring ceremony, first dance and cutting wedding cake). This might be possible for certain events where a clearly defined storyline is well known, such definition might not be readily available for the vast majority of videos. Secondly, compared with manually generating a summary video for a given input video sequence, identifying separately each component in the storyline is a much more challenging task for users, and it sometimes even requires expert knowledge on the event depicted in the collection of videos. For example, for non-experts of figure skating, it might be difficult to tell the difference between Axel jump, Lutz jump and Flip jump, but it is rather common sense that they should all be incorporated into the summary video as they are the exciting moments in a figure skating performance. On the other hand, if the ground truth event annotation treats those segments as the same event (e.g. Jump), the internal difference between those segments could significantly hinder the accuracy of event recognition and hence affect the performance of video summarization. Finally, separately identifying each component in the storyline will result in multiple event recognizers, which could significantly slow the summarization of unseen testing videos. Specifically, to summarize an unseen video, each event recognizer needs at least one pass through the video, resulting in multiple passes through the testing video for summarization. With a complicated storyline with large number of components, those multiple passes could render the algorithm impractical.

Weighing the above considerations, in this chapter, we use a much weaker form of supervision: instead of asking users to separately label out each component of the storyline, we only need users to generate their preferred summary video, with no requirement on assigning semantic label to each component in the summary. Specifically, for the wedding video example, the only supervised information required is which segments should be incorporated into the summary, without any knowledge on which segment shows bride walking down the aisle, or which segment corresponds to the ring ceremony. Even more interestingly, the total number of unique events in the storyline is also unknown. Though lacking semantic information on each segment incorporated into the summary video, our formulation of supervised video summarization

is more practical and applicable to a much wider collection of videos.

One clear challenge in our formulation of supervised video summarization lies in the lack of knowledge on the number of unique events in the storyline. If the number of unique events is set too large, it could mean that each individual segment in the summary video is treated as a unique event, and the learned classifier cannot discover the similarities among multiple appearances of the same event (such as bride walking down the aisle segments from different wedding videos), rendering the supervised video summarization algorithm into a simple pattern matching trick, with rather weak generalization ability to unseen videos. On the other hand, if the number of unique events is set too small, various types of segments could end up being treated as the same event. The significant internal difference between those segments could make learning classifier very challenging and hinder the performance of later summarization. Therefore, how to properly select the number of unique events in the summary video automatically is an important yet challenging problem. Moreover, while some videos present the entire sequence of certain events, others might only have partial events. For example, some videos could contain the entire event of couple cutting the wedding cake and sharing a price together, others might only show the cake without the cutting and sharing part. If the learned video summarization model can only recognize full event, it will miss similar events with only partial segments in the testing video, hence resulting in incomplete summary video. Therefore, the algorithm should understand not only entire sequence, but also partial sequences. In the rest of this section, we will provide details on formulating supervised video summarization into a max margin classification problem.

7.2.1 Problem Formulation

Let $(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)$ be the set of training video sequences and their associated ground truth annotations for the summary. Specifically, suppose in video $\mathbf{X}_i$, there are m_i video segments incorporated into the summary, then we have $\mathbf{Y}_i = \cup_{k=1}^{m_i} \mathbf{y}_i^k$ where $\mathbf{y}_i^k = [s_i^k, e_i^k]$ consists of two numbers indicating the start and the end of i-th component in the desired summary video for $\mathbf{X}_i$. For any interval $\mathbf{y} = [s, e]$, define $\mathbf{X}_\mathbf{y}$ as the segment of $\mathbf{X}$ from frame s to e. Moreover, define $\Psi(\mathbf{X}_\mathbf{y})$ as the output of video summarizer, which has positive response for those segments included in the summary video, and negative values for other excluded segments, i.e., $\Psi(\mathbf{X}_\mathbf{y}) > 0$ for any $\mathbf{y} \subset \mathbf{Y}$ and $\Psi(\mathbf{X}_\mathbf{y}) < 0$ for any $\mathbf{y} \not\subset \mathbf{Y}$. In testing, the learned video summarizer Ψ is applied to any video segment from the testing video $\mathbf{X}$ and output summary will be computed as $\mathbf{Y} = \cup_{\mathbf{y}:\Psi(\mathbf{X}_\mathbf{y})>0} \mathbf{y}$. To avoid trivial selection of too short or too long video segments, we further constrain bound the length of segments using l_{min} and l_{max} such that only segment with length $l_{min} \leq |\mathbf{y}| \leq l_{max}$ will be tested using the learned video summarizer and $\mathbf{Y} = \cup_{\mathbf{y}:l_{min}\leq|\mathbf{y}|\leq l_{max},\Psi(\mathbf{X}_\mathbf{y})>0} \mathbf{y}$.

7.2.2 Augmenting the Training Data

In the above formulation for supervised video summarization, a key assumption is that both the training videos and testing videos are recording events of similar nature, such as a collection of wedding videos. It is the similar nature between training data and testing data that enables the use of the manually generated storyline for training videos to summarize testing video. However, a key challenge in such formulation is the internal variation among different videos, though they

Figure 7.2: (Best viewed in color) Augmenting the positive training examples with simulated partial events. The 15 frames shown here represent a complete event of *"cutting wedding cake"* from a wedding video. Besides, 3 partial events are also shown using the red, blue and green boxes. It should be noted that much more partial events are simulated in our max margin video summarization formulation, and the simulated partial events are allowed to overlap.

are recording events of similar nature. In the wedding video example, the training videos could show the complete sequence of couple cutting the wedding cake and sharing a piece together, while the testing video might only have partial sequence showing the cutting part. If our trained model can only recognize complete event, it might not identify the partial occurrence of the similar events in the testing video, rendering the summary video missing crucial component.

To enable the learned video summarizer to recognize both complete event and various possible partial events, we propose to augment the training data by introducing simulated partial events into the set of positive examples [58]. Figure 7.2 illustrates our idea of augmenting the training data. Specifically, given training video sequence $\mathbf{X}$ and its associated ground truth annotation for the summary $\mathbf{Y} = \cup_{k=1}^{m} \mathbf{y}^k$, where each $\mathbf{y}^k$ is a separate component in the manually generated summary. For each event $\mathbf{y}^k$, we simulate the following set of partial events contained in $\mathbf{y}^k$ with bounded length

$$\tilde{\mathbf{y}}^k = \{\mathbf{y} | \mathbf{y} \subset \mathbf{y}^k, l_{min} \leq |\mathbf{y}| \leq l_{max}\} \tag{7.1}$$

Consequently, we define $\tilde{\mathbf{Y}}$ as the augmented summary containing both the components in $\mathbf{Y}$ and augmented partial events, i.e., $\tilde{\mathbf{Y}} = \cup_{k=1}^{m} \tilde{\mathbf{y}}^k$. The decision function Ψ should therefore generate positive response to not only the original complete events, but also simulated partial events, i.e., $\Psi(\mathbf{X}_\mathbf{y}) > 0$ for any $\mathbf{y} \subset \tilde{\mathbf{Y}}$.

7.2.3 Max Margin Video Summarization

Let $\mathbf{g}(\mathbf{X}_\mathbf{y})$ be the feature vector for video segment $\mathbf{X}_\mathbf{y}$ (details on $\mathbf{g}(\cdot)$ will be discussed in the experiments section). For each training video $\mathbf{X}_\mathbf{y}$, multiple types of events are included in the

ground truth summary $\mathbf{Y}$, such as bride walking down the aisle, ring ceremony, cutting the wedding cake, etc., in the wedding video example. In max margin video summarization, we define a separate linear decision function for each type of such events. Assume the total number of unique events in the collection of videos recording events of similar nature is K, *GuideSum* defines K linear decision functions $\{f(\mathbf{X}_\mathbf{y}; \mathbf{w}_k, b_k)\}_{k=1}^K$ with $f(\mathbf{X}_\mathbf{y}; \mathbf{w}_k, b_k) = \mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k$. Then for positive training examples, at least one of $\{f(\mathbf{X}_\mathbf{y}; \mathbf{w}_k, b_k)\}_{k=1}^K$ should generate positive response. On the other hand, for negative training examples, all $\{f(\mathbf{X}_\mathbf{y}; \mathbf{w}_k, b_k)\}_{k=1}^K$ will have negative values. Therefore, the decision function $\Psi(\mathbf{X}_\mathbf{y})$ of *GuideSum* could be defined as follows:

$$\Psi(\mathbf{X}_\mathbf{y}) = \max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k\right) \tag{7.2}$$

As we discussed early in this section, a key challenge in max margin video summarization is properly setting the number of unique events K in the collection of videos. Specifically, we would like a learning method that facilitates both induction of new event type and removal of event types with low predictive power. To achieve this goal, we apply ℓ_1/ℓ_2 regularization on the set of weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$. The idea for this regularization is that uninformative weight vectors will be regularized towards $\mathbf{0}$, effectively removing it from the set of decision functions. Therefore, max margin video summarization is formulated as the following optimization problem:

$$\min_{\{\mathbf{w}_k, b_k\}_{k=1}^K} \quad \sum_{k=1}^K \|\mathbf{w}_k\|_2 + C\left(\sum_{i=1}^n \sum_{\mathbf{y}_i^p \in \mathbf{Y}_i} \xi_i^p + \sum_{i=1}^n \delta_i\right) \tag{7.3}$$

$$s.t. \quad \forall i = 1, \ldots, n, \forall \mathbf{y}_i^p \subset \mathbf{Y}_i, \forall \mathbf{y} \subset \tilde{\mathbf{y}}_i^p, \ \max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k\right) \geq 1 - \xi_i^p \frac{|\mathbf{y}_i^p|}{|\mathbf{y}|} \tag{7.4}$$

$$\forall i = 1, \ldots, n, \forall \mathbf{y} \not\subset \tilde{\mathbf{Y}}_i, \forall k = 1, \ldots, K, \ -(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k) \geq 1 - \delta_i \tag{7.5}$$

where the superscript p corresponds to positive training examples, and all video segments $\mathbf{y}$ considered in constraints (7.4) and (7.5) satisfy the length constraint $l_{min} \leq |\mathbf{y}| \leq l_{max}$. In constraint (7.4), each slack variable ξ_i^p is scaled by the fraction of partial event $\mathbf{y}$ in the complete event $\mathbf{y}_i^p$. This will effectively enforce a tighter constraint on complete event, while the shorter the partial event, the looser the constraint will be. Algorithm 5 summarizes our proposed max margin video summarization algorithm *GuideSum*.

7.3 Optimization

The difficulty in optimization problem (7.3) lies in the following three aspects: (1) the constraints in (7.4) are non-convex; (2) augmentation on training data results in huge number of constraints in (7.4) and (7.5); (3) the ℓ_1/ℓ_2 regularization in the objective function. In this section, we provide details on solving problem (7.3), where the non-convexity of constraints is handled by *Constrained Concave-Convex Procedure (CCCP)*, we then use *Cutting Plane algorithm* to reduce the huge number of constraints into a much smaller subset, and the ℓ_1/ℓ_2 regularized problem is efficiently solved by *Alternating Direction Method of Multipliers (ADMM)*.

Algorithm 5 Guided Video Summarization (GuideSum)

input Training videos and associated ground truth annotations for the summary $(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)$; Testing video $\mathbf{X}$

output Short video $\mathbf{Y}$ summarizing storyline of testing video $\mathbf{X}$

1: Solve optimization problem (7.3) for optimal parameters $\{\mathbf{w}_1, \ldots, \mathbf{w}_K, b_1, \ldots, b_K\}$ of the decision function $\Psi(\cdot)$, and initialize $\mathbf{Y} = \emptyset$.
2: **for all** video segments $\mathbf{X}_\mathbf{y} \subset \mathbf{X}$ with $l_{min} \leq |\mathbf{y}| \leq l_{max}$ **do**
3: Compute decision function $\Psi(\mathbf{X}_\mathbf{y})$ as (7.2)
4: **if** $\Psi(\mathbf{X}_\mathbf{y}) > 0$ **then**
5: Incorporate $\mathbf{y}$ into summary video $\mathbf{Y} = \mathbf{Y} \cup \mathbf{y}$
6: **end if**
7: **end for**

7.3.1 Constrained Concave-Convex Procedure

The *concave-convex procedure* [140] is a method for solving non-convex optimization problem whose objective function could be expressed as a difference of convex functions. It can be viewed as a special case of variational bounding [63] and related techniques including lower(upper) bound maximization(minimization) [86], surrogate functions and majorization [72]. While in [140] the authors only considered linear constraints, [119] proposed a generalization, the *constrained concave-convex procedure* (*CCCP*), for problems with a concave-convex objective function under concave-convex constraints.

Assume we are solving the following optimization problem [119]

$$\begin{aligned} \min_{\mathbf{z}} \quad & f_0(\mathbf{z}) - g_0(\mathbf{z}) \\ s.t. \quad & f_i(\mathbf{z}) - g_i(\mathbf{z}) \leq c_i \quad i = 1, \ldots, n \end{aligned} \tag{7.6}$$

where f_i and g_i are real-valued convex functions on a vector space $\mathcal{Z}$ and $c_i \in \mathcal{R}$ for all $i = 1, \ldots, n$. Denote by $T_1\{f, \mathbf{z}\}(\mathbf{z}')$ the first order *Taylor expansion* of f at location $\mathbf{z}$, that is $T_1\{f, \mathbf{z}\}(\mathbf{z}') = f(\mathbf{z}) + \partial_\mathbf{z} f(\mathbf{z})(\mathbf{z}' - \mathbf{z})$, where $\partial_\mathbf{z} f(\mathbf{z})$ is the gradient of the function f at $\mathbf{z}$. For non-smooth functions, it can be easily shown that the gradient $\partial_\mathbf{z} f(\mathbf{z})$ would be replaced by the subgradient [28]. Given an initial point $\mathbf{z}_0$, the *CCCP* computes $\mathbf{z}_{t+1}$ from $\mathbf{z}_t$ by replacing $g_i(\mathbf{z})$ with its first-order Taylor expansion at $\mathbf{z}_t$, i.e. $T_1\{g_i, \mathbf{z}_t\}(\mathbf{z})$, and setting $\mathbf{z}_{t+1}$ to the solution to the following relaxed optimization problem

$$\begin{aligned} \min_{\mathbf{z}} \quad & f_0(\mathbf{z}) - T_1\{g_0, \mathbf{z}_t\}(\mathbf{z}) \\ s.t. \quad & f_i(\mathbf{z}) - T_1\{g_i, \mathbf{z}_t\}(\mathbf{z}) \leq c_i \quad i = 1, \ldots, n \end{aligned} \tag{7.7}$$

The above procedure continues until $\mathbf{z}_t$ converges, and [119] proved that the *CCCP* is guaranteed to converge to local optimum.

Specifically, the objective function in (7.3) is convex, the constraint (7.4) is, though non-convex, a difference between two convex functions. Hence, we can solve problem (7.3) with the *constrained concave-convex procedure*. Notice that while $\max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_y) + b_k\right)$ is convex, it

is a non-smooth function of $\{\mathbf{w}_k, b_k\}_{k=1}^K$. To use the *CCCP*, we need to replace the gradients by the *subgradients* [28]:

$$\partial_{\mathbf{w}_k} \max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_y) + b_k\right)\Big|_{\mathbf{w}_k=\mathbf{w}_k^t} = \begin{cases} \mathbf{g}(\mathbf{X}_y) & \text{if } k = \arg\max_p \left(\mathbf{w}_p^{t\top} \mathbf{g}(\mathbf{X}_y) + b_p^t\right) \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

$$\partial_{b_k} \max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_y) + b_k\right)\Big|_{b_k=b_k^t} = \begin{cases} 1 & \text{if } k = \arg\max_p \left(\mathbf{w}_p^{t\top} \mathbf{g}(\mathbf{X}_y) + b_p^t\right) \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

Given an initial point $\{\mathbf{w}_k^0, b_k^0\}_{k=1}^K$, define $k^* = \arg\max_k \left(\mathbf{w}_k^{t\top} \mathbf{g}(\mathbf{X}_y) + b_k^t\right)$, the *CCCP* computes $\{\mathbf{w}_k^{t+1}, b_k^{t+1}\}_{k=1}^K$ from $\{\mathbf{w}_k^t, b_k^t\}_{k=1}^K$ by replacing $\max_k \left(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_y) + b_k\right)$ in the constraint with its first-order Taylor expansion at $\{\mathbf{w}_k^t, b_k^t\}_{k=1}^K$, i.e.

$$\max_k \left(\mathbf{w}_k^{t\top} \mathbf{g}(\mathbf{X}_y) + b_k^t\right) + \mathbf{g}(\mathbf{X}_y)^\top (\mathbf{w}_{k^*} - \mathbf{w}_{k^*}^t) + (b_{k^*} - b_{k^*}^t) = \mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_y) + b_{k^*} \tag{7.10}$$

By substituting the above first-order Taylor expansion (7.10) into problem (7.3), we obtain the following:

$$\min_{\{\mathbf{w}_k, b_k\}_{k=1}^K} \quad \sum_{k=1}^K \|\mathbf{w}_k\|_2 + C \left(\sum_{i=1}^n \sum_{\mathbf{y}_i^p \in \mathbf{Y}_i} \xi_i^p + \sum_{i=1}^n \delta_i \right) \tag{7.11}$$

$$s.t. \quad \forall i = 1, \ldots, n, \forall \mathbf{y}_i^p \subset \mathbf{Y}_i, \forall \mathbf{y} \subset \tilde{\mathbf{y}}_i^p,\ \mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_{k^*} \geq 1 - \xi_i^p \frac{|\mathbf{y}_i^p|}{|\mathbf{y}|} \tag{7.12}$$

$$\forall i = 1, \ldots, n, \forall \mathbf{y} \not\subset \tilde{\mathbf{Y}}_i, \forall k = 1, \ldots, K,\ -(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k) \geq 1 - \delta_i \tag{7.13}$$

where $k^* = \arg\max_k \left(\mathbf{w}_k^{t\top} \mathbf{g}(\mathbf{X}_y) + b_k^t\right)$. Following the *CCCP*, the obtained solution $\{\mathbf{w}_k, b_k\}_{k=1}^K$ from problem (7.11) is then used as $\{\mathbf{w}_k^{t+1}, b_k^{t+1}\}_{k=1}^K$ and the iteration continues until convergence. Specifically, Algorithm 6 provides details on using *CCCP* to solve problem (7.3).

Algorithm 6 Solve Problem (7.3) Using *CCCP*

Initialize $\{\mathbf{w}_k^0, b_k^0\}_{k=1}^K$ with random values
repeat
  Find $\{\mathbf{w}_k^{t+1}, b_k^{t+1}\}_{k=1}^K$ as the solution to problem (7.11)
until stopping criterion satisfied [119]

7.3.2 Cutting-Plane Algorithm

In each iteration of *CCCP*, we need to solve problem (7.11). Due to the augmentation of training data by introducing simulated partial events as positive examples, and the large collection of negative examples, the huge number of constraints in problem (7.11) poses great challenge. In this section, we propose to use *cutting plane algorithm*, which targets to find a small subset of constraints from the whole set of constraints in problem (7.11) that ensures a sufficiently accurate solution. Specifically, the *cutting plane altorithm* [64, 130] constructs a nested sequence of successively tighter relaxations of problem (7.11), and has been theoretically proven to find a

polynomially sized subset of constraints, with which the solution of the relaxed problem fulfills all constraints from problem (7.11) up to a precision of ϵ. That is to say, the remaining exponential number of constraints are guaranteed to be violated by no more than ϵ, without the need for explicitly adding them to the optimization problem [130]. The *cutting plane algorithm* starts with an empty constraint subset Ω, and it computes the optimal solution to problem (7.11) subject to the constraints in Ω. The algorithm then finds the most violated constraint in problem (7.11) and adds it into the subset Ω. In this way, we construct a successive strengthening approximation of the original problem by a cutting plane that cuts off the current optimal solution from the feasible set [64]. The algorithm stops when no constraint in (7.11) is violated by more than ϵ.

Here, the feasibility of a constraint is measured by the corresponding value of ξ_i^p and δ_i, therefore, the most violated constraint is the one that would result in the largest values for those slack variables. Specifically, for each training video $(\mathbf{X}_i, \mathbf{Y}_i)$, with m_i separate components $\{\mathbf{y}_i^p\}_{p=1}^{m_i}$ in the ground truth annotation $\mathbf{Y}_i$, we define m_i constraint subsets Ω_i^p and Ω_i^-. Then for any $\mathbf{y}_i^p \in \mathbf{Y}$, the most violated constraint from (7.12) is the one that would result in the largest ξ_i^p. In order to fulfill all constraints in problem (7.12), the minimum value of ξ_i^p is as follows

$$\xi_i^{p*} = \max_{\mathbf{y} \subset \tilde{\mathbf{y}}_i^p} \frac{|\mathbf{y}|}{|\mathbf{y}_i^p|} \left[1 - (\mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_{k^*})\right] \tag{7.14}$$

Therefore, the most violated constraint to be added into Ω_i^p is the one causing $\xi_i^p = \xi_i^{p*}$. Similarly, the most violated constraint to be added into Ω_i^- is the one causing δ_i to reach the following value

$$\delta_i^* = \max_{k, \mathbf{y} \not\subset \tilde{\mathbf{Y}}_i} \left[1 + (\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k)\right] \tag{7.15}$$

Assume the current working constraint sets are $\{\{\Omega_i^p\}_{p=1}^{m_i}, \Omega_i^-\}_{i=1}^n$, *cutting plane algorithm* solves the following problem

$$\min_{\{\mathbf{w}_k, b_k\}_{k=1}^K} \quad \sum_{k=1}^K \|\mathbf{w}_k\|_2 + C \left(\sum_{i=1}^n \sum_{\mathbf{y}_i^p \in \mathbf{Y}_i} \xi_i^p + \sum_{i=1}^n \delta_i \right) \tag{7.16}$$

$$s.t. \quad \forall i = 1, \ldots, n, \forall \mathbf{y}_i^p \subset \mathbf{Y}_i, \forall \mathbf{y} \in \Omega_i^p, \ \mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_{k^*} \geq 1 - \xi_i^p \frac{|\mathbf{y}_i^p|}{|\mathbf{y}|} \tag{7.17}$$

$$\forall i = 1, \ldots, n, \forall (k, \mathbf{y}) \in \Omega_i^-, \ -(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k) \geq 1 - \delta_i \tag{7.18}$$

Algorithm 7 summarizes the *cutting plane algorithm* for solving problem (7.11).

7.3.3 Alternating Direction Method of Multipliers

To solve problem (7.16) under working constraint set $\{\{\Omega_i^p\}_{p=1}^{m_i}, \Omega_i^-\}_{i=1}^n$, we first reformulate it as follows

$$\min_{\{\mathbf{w}_k, b_k, \mathbf{z}_k\}_{k=1}^K} \quad \sum_{k=1}^K \|\mathbf{z}_k\|_2 + C \left(\sum_{i=1}^n \sum_{\mathbf{y}_i^p \in \mathbf{Y}_i} \xi_i^p + \sum_{i=1}^n \delta_i \right) + \mathcal{I}\left(\{\mathbf{w}_k, b_k\}_{k=1}^K \in \mathbf{\Gamma}\right) \tag{7.21}$$

$$s.t. \quad \forall k = 1, \ldots, K: \ \mathbf{w}_k - \mathbf{z}_k = \mathbf{0} \tag{7.22}$$

Algorithm 7 Solve Problem (7.11) Using *Cutting Plane* Algorithm

Initialize $\Omega_i^p = \Omega_i^- = \emptyset$, and set precision parameter ϵ
repeat
  Solve problem (7.16) under the current working constraint set $\{\{\Omega_i^p\}_{p=1}^{m_i}, \Omega_i^-\}_{i=1}^n$
  for $i = 1, \ldots, n$; $p = 1, \ldots, m_i$ **do**
    Select the most violated constraint using (7.14)
    if the selected constraint is violated by more than ϵ:

$$\mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_{k^*} < 1 - (\xi_i^p + \epsilon)\frac{|\mathbf{y}_i^p|}{|\mathbf{y}|} \tag{7.19}$$

    then
      Add the selected constraint into Ω_i^p
    end if
  end for
  for $i = 1, \ldots, n$ **do**
    Select the most violated constraint using (7.15)
    if the selected constraint is violated by more than ϵ:

$$-(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k) < 1 - (\delta_i + \epsilon) \tag{7.20}$$

    then
      Add the selected constraint into Ω_i^-
    end if
  end for
until no constraint is violated by more than ϵ

where $\mathbf{\Gamma}$ is the set of feasible $\{\mathbf{w}_k, b_k\}_{k=1}^K$ for constraints (7.17) and (7.18), $\mathcal{I}$ is indicator function such that $\mathcal{I}(\{\mathbf{w}_k, b_k\}_{k=1}^K \in \mathbf{\Gamma}) = 0$ if $\{\mathbf{w}_k, b_k\}_{k=1}^K$ satisfies constraints (7.17) and (7.18), and $\mathcal{I}(\{\mathbf{w}_k, b_k\}_{k=1}^K \in \mathbf{\Gamma}) = +\infty$ otherwise. Then *ADMM* solves problem (7.21) using the following iterations:

$$\{\mathbf{w}_k^{t+1}, b_k^{t+1}\}_{k=1}^K = \underset{\{\mathbf{w}_k, b_k\}_{k=1}^K}{\arg\min} \left(f(\{\mathbf{w}_k, b_k\}_{k=1}^K) + \frac{\rho}{2}\sum_{k=1}^K \|\mathbf{w}_k - \mathbf{z}_k^t + \mathbf{u}_k^t\|_2^2 \right) \tag{7.23}$$

$$\mathbf{z}_k^{t+1} = \mathcal{S}_{1/\rho}(\mathbf{w}_k^{t+1} + \mathbf{u}_k^t),\ \forall k = 1, \ldots, K \tag{7.24}$$

$$\mathbf{u}_k^{t+1} = \mathbf{u}_k^t + \mathbf{w}_k^{t+1} - \mathbf{z}_k^{t+1},\ \forall k = 1, \ldots, K \tag{7.25}$$

where $f(\{\mathbf{w}_k, b_k\}_{k=1}^K)$ is defined as

$$f(\{\mathbf{w}_k, b_k\}_{k=1}^K) = C\left(\sum_{i=1}^n \sum_{\mathbf{y}_i^p \in \mathbf{Y}_i} \xi_i^p + \sum_{i=1}^n \delta_i\right) + \mathcal{I}\left(\{\mathbf{w}_k, b_k\}_{k=1}^K \in \mathbf{\Gamma}\right) \tag{7.26}$$

and $\mathcal{S}$ is the soft-thresholding operator defined as

$$\mathcal{S}_\kappa(a) = \max\{(1 - \kappa/|a|)a, 0\} \tag{7.27}$$

In the above *ADMM* iterations, both $\mathbf{z}_k$ update (7.24) and $\mathbf{u}_k$ update (7.25) are trivial to compute. The $\{\mathbf{w}_k, b_k\}$ update in (7.23) is equivalent to the following constrained optimization problem

$$\min_{\{\mathbf{w}_k, b_k\}_{k=1}^K} \quad \frac{\rho}{2}\sum_{k=1}^{K}\|\mathbf{w}_k - \mathbf{z}_k^t + \mathbf{u}_k^t\|_2^2 + C\left(\sum_{i=1}^{n}\sum_{\mathbf{y}_i^p \in \mathbf{Y}_i}\xi_i^p + \sum_{i=1}^{n}\delta_i\right) \tag{7.28}$$

$$s.t. \quad \forall i = 1, \ldots, n, \forall \mathbf{y}_i^p \subset \mathbf{Y}_i, \forall \mathbf{y} \in \Omega_i^p,\ \mathbf{w}_{k^*}^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_{k^*} \geq 1 - \xi_i^p \frac{|\mathbf{y}_i^p|}{|\mathbf{y}|} \tag{7.29}$$

$$\forall i = 1, \ldots, n, \forall (k, \mathbf{y}) \in \Omega_i^-,\ -(\mathbf{w}_k^\top \mathbf{g}(\mathbf{X}_\mathbf{y}) + b_k) \geq 1 - \delta_i \tag{7.30}$$

Comparing the above problem with (7.16), we can see that *ADMM* effectively reduces an ℓ_1/ℓ_2 regularized problem into a series of ℓ_2 regularized problems. Specifically, problem (7.28) is a conventional SVM-type optimization problem, and can be efficiently solved using *Stochastic Gradient Descent* [19].

Finally, since both *cutting plane algorithm* and *ADMM* are guaranteed to converge to global optimum, while *CCCP* guarantees convergence to local optimum, our proposed optimization algorithm for *GuideSum* is guaranteed to converge to local optimum of problem (7.3).

7.4 Experiments

This section describes our experiments on various videos, including synthetic data, surveillance videos, and YouTube videos for various scenarios.

7.4.1 Feature Representations

We adopt the same feature representation for video data as used in Chapter 5. Specifically, we utilize the spatio-temporal interest points detected using the method in [40], and describe each detected interest point with histogram of gradient (HoG) and histogram of optical flow (HoF). The feature representation for each detected interest point is then obtained by concatenating the HoG feature vector and HoF feature vector. Finally, feature vectors for detected interest points in the training videos are clustered using *k-means* to create a codebook of D words (the value of D is determined according to the complexity of the videos). Subsequently, each interest point is represented by the ID of the corresponding codebook entry, and each video segment is represented by the histogram of words associated with interested points inside the segment.

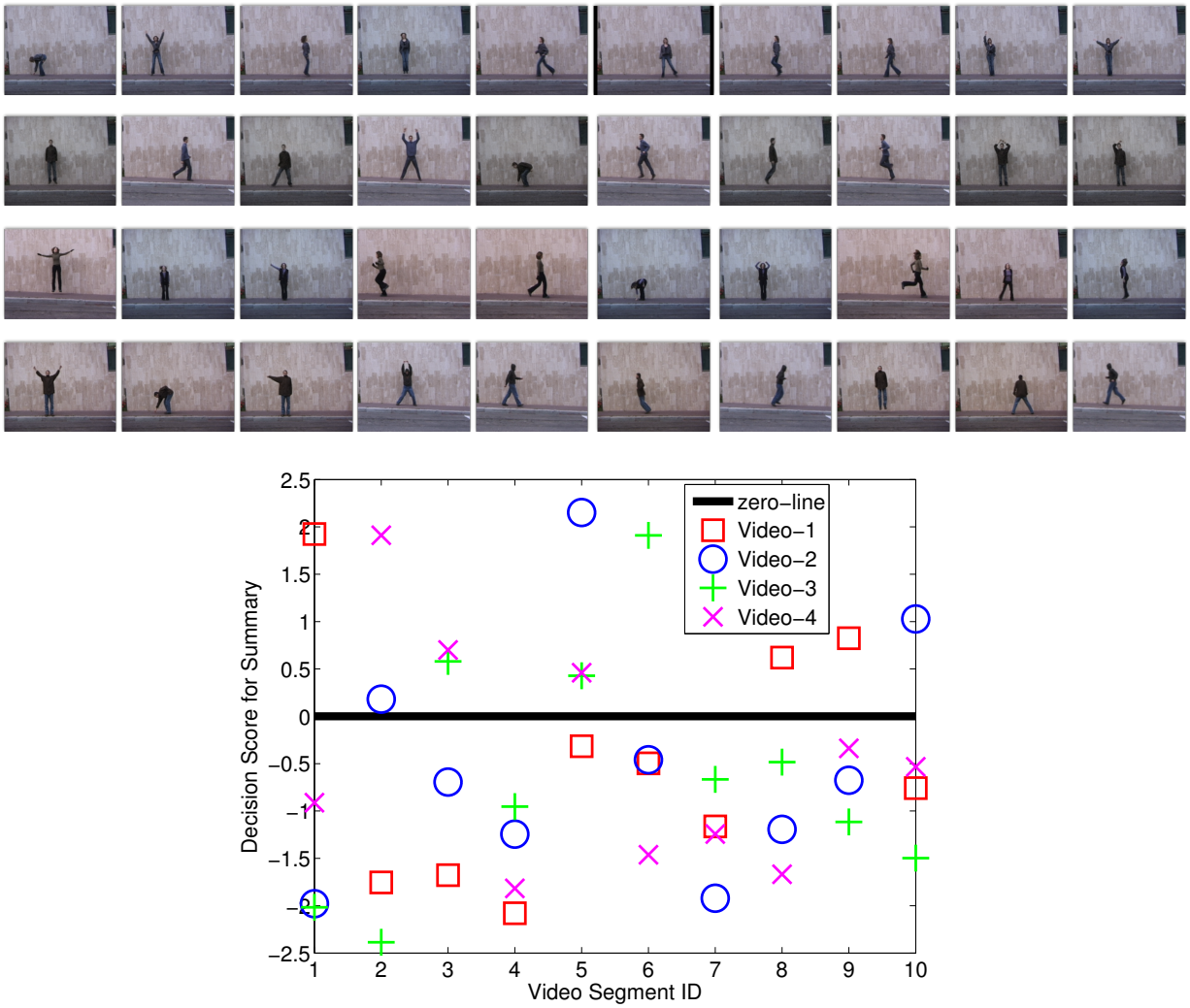

Figure 7.3: (Up) 4 example synthetic videos (used as testing data in our experiment), each constructed using 10 videos of different action classes performed by the same person; (Down) Decision score for each video segment in the 4 testing videos. Specifically, Video 1 is constructed as: {B, K, J, P, R, G, S, W, O, T}; Video 2 is constructed as {P, W, G, K, B, R, J, S, T, O}; Video 3 is constructed as {K, P, O, S, W, B, T, R, S, J}; Video 4 is constructed as {T, B, O, K, W, J, S, P, S, R}. As shown in the Down figure, *GuideSum* correctly identifies *walk*, *bend* and *one-hand-wave* segments from all 4 testing videos, hence generating the correct summary videos. Specifically, for Video-1, the segments with positive decision scores are segment 1 (*bend*), segment 8 (*walk*), and segment 9 (*one-hand-wave*).

7.4.2 Synthetic Videos

We first validate the performance of *GuideSum* on a synthetically generated data set of 9 video sequences. Specifically, we use videos from the ten different action categories from *Weizmann* action recognition data [53], i.e., *walk (W)*, *run (R)*, *jump (J)*, *gallop sideways (G)*, *bend (B)*, *one-hand-wave (O)*, *two-hands wave (T)*, *jump in place (P)*, *jumping back (K)* and *skip (S)*. For each of the ten action classes, 9 different people are asked to perform the action and record the video. The 9 videos in the synthetic data set are constructed by concatenating all action videos performed by the same person, in random order. Example video sequences are shown in Figure 7.3. To formulate the supervised video summarization problem, we identify 3 action classes, i.e., *walk*, *bend*, *one-hand-wave*, as the events included in the ground truth annotation for summary videos. We randomly select 5 video sequences as training data, and test on the rest 4 synthetic videos. Figure 7.3 shows the decision score for all 4 testing videos, where *GuideSum* clearly generates the correct summary videos, including only video segments corresponding to *walk*, *bend*, and *one-hand-wave*, i.e., the 3 action classes identified as components of the ground truth summary video. Moreover, in the experiment, we set $K = 10$, much larger than the correct number of action types 3. More interestingly, in the summarization model learned by *GuideSum*, the number of non-zero weight vectors in $\{\mathbf{w}_k\}_{k=1}^{10}$, is equal to the correct number of action types in the ground truth annotation for summary videos, i.e., 3. Therefore, *GuideSum* not only generates the correct summary for testing videos, but also correctly identifies the true number of unique "components" in the summary video.

7.4.3 Surveillance Videos

We collect 3 videos recorded by a surveillance camera monitoring a shopping mall [1]. Important events which should be incorporated into the summitry video include: (1) person running fast in the mall; (2) person holding a strange sign near the surveillance camera.

The same performance measure used in Chapter 6 is adopted in this section. Specifically, for each video in our data set, three judges selected segments from original video to compose their preferred version of summary video. The final ground truth is then constructed by pooling together those segments selected by at least two judges. Following [30], to quantitatively determine the overlap between algorithm generated summary and ground truth, both video segment content and time differences are considered. More precisely, we augment the ground truth summary video via potentially extending each segment by 2 seconds before and after it. Specifically, the ground truth summary video is composed by a series of video segments from the original video. For the i-th video segment in the ground truth summary video, we look at the video contents 2 seconds before it. If this 2 seconds of video contents show similar scene content and motion pattern as the i-th video segment, we add it into the augmented ground truth. We perform the same procedure for the video contents 2 seconds after the i-th video segment. Consequently, the final augmented ground truth summary video contains the original user generated ground truth summary video, and those 2 seconds of video contents which are showing similar scene content and motion pattern as their corresponding ground truth video segment. Final accuracy is computed as the ratio of the number of overlapped frames between the augmented ground truth summary video and algorithm generated summary video, divided by the length of the original ground truth

Table 7.1: Length of both the original and summary videos (seconds) for 3 surveillance videos used in experiments.

Surveillance Video	#1	#2	#3
Original	3326	2399	3621
Summary	89.7	88.4	92.0

summary video. To enable fair comparison across several video summarization algorithms, parameters for various algorithms are set such that the length of generated summary videos are the same as ground truth summary video. For example, in *GuideSum*, we rank the decision scores for all video segments in the testing video, and select the top ranked video segments, whose combined length is the same as the ground truth annotation for summary, to construct the summary video. Details on the length of original and summary videos are provided in Table 7.1.

We compare *GuideSum* with several other methods. Due to the lack of video summarization algorithms capable of utilizing supervised information, we compare against several popular unsupervised video summarization methods, including *evenly spaced segments*, *K-means clustering* [30], the *DSVS* algorithm proposed in [30], and *LiveLight* [145], state-of-the-art method for unsupervised video summarization. It is shown in [30] that *DSVS* already beats color-histogram based method [109] and motion-based method [85]. The above unsupervised video summarization algorithms are set up the same way as in Chapter 6. Specifically, we first apply temporal segmentation of the original video, evenly divided into segments each with a constant length of 50 frames. *Evenly spaced segments* computes the number of segments N via dividing the length of ground truth summary video by the length of each video segment, i.e., 50 frames, then select N segments starting from the first segment with even space between any two segments. Those evenly spaced segments are then concatenated to compose the summary video for the *evenly spaced segments* method. For the *K-means clustering* method, we first group the collection of video segments from the original video into N clusters, using *K-means clustering*, where N is computed the same way as in the *evenly spaced segments* method. Then for each center of the N clusters, we find the segment from the original video that has minimum distance to that cluster center. Finally, the summary video for *K-means clustering* method is composed by concatenating the N video segments with minimum distance to the N cluster centers. For the *DSVS* algorithm, we use the 50-frame video segments as basic units in the algorithm. Besides the above collection of unsupervised video summarization algorithms, our problem formulation also bares similarity to the problem of event detection / recognition. Therefore, we also compare against state-of-the-art event detection method, *max margin early event detection (MMED)* [58]. Due to the lack of differentiation among different types of events, *MMED* has to treat all video segments identified in the ground truth annotation as the target event. The intrinsic difference between video segments in the ground truth summary video could pose great challenge to the performance of *MMED*. On the other hand, our *GuideSum* is designed to handle different types of events in the ground truth summary videos. For both *GuideSum* and *MMED*, we use 2 videos and their associated ground truth annotation as training data, and test on the 3rd video. For all algorithms, the reported accuracy is the averaged score across 3 videos in the data set.

According to the quantitative comparison provided in Table 7.2, we see that *GuideSum*

Table 7.2: Accuracy comparison (%) across various algorithms on surveillance videos and YouTube videos.

	EvenSpace	K-Means	DSVS	LiveLight	MMED	GuideSum
Surveillance	39.94	59.22	85.04	93.36	96.26	**97.84**
Wedding	41.99	32.78	44.10	52.38	66.07	**83.43**
FigureSkating	23.65	28.61	51.44	56.73	72.30	**86.79**
CarRacing	38.01	46.92	59.33	61.42	71.27	**82.45**

achieves higher accuracy than the entire collection of unsupervised video summarization algorithms. This should not come as a surprise, since *GuideSum* could effectively utilize the supervised information provided in ground truth annotation for summary video in the training data. Moreover, *GuideSum* is slightly more accurate than the state-of-the-art event detection method *MMED*. The relatively small margin between *GuideSum* and *MMED* is due to the simplicity of the surveillance videos, as the events incorporated into the ground truth summary video are mainly corresponding to people running fast in the mall, hence reducing the complicated summarization problem into almost a single event detection problem.

7.4.4 Youtube Videos

Besides surveillance videos, with fixed camera and relatively controlled environment, a more interesting application of *GuideSum* is summarizing YouTube videos. In this section, we apply our method to 3 scenarios of videos: (1) wedding videos; (2) figure skating videos; and (3) car racing videos. Our motivation for selecting those 3 groups of videos is the clear definition of storyline or exciting moments. For example, in wedding videos, there is a clear storyline as already discussed in previous sections; in figure skating videos, a clear definition of exciting moments include the various types of spinning moves and jumping stunts; for car racing videos, the storyline and exciting moments include starting, passing, pit stop and finishing.

For each of the 3 scenarios, we collect 10 videos from YouTube, and acquire ground truth annotation for summary video in the same fashion as in previous section on surveillance videos. For each scenario, in *GuideSum* and *MMED*, we randomly split the 10 videos into two sets $\mathbf{A}$ and $\mathbf{B}$, each containing 5 videos. We first train on set $\mathbf{A}$, and test on $\mathbf{B}$, then reverse the order, i.e., train on $\mathbf{B}$ and test on $\mathbf{A}$. Average accuracy over the entire 10 videos (each video is used as testing data once) is calculate, and compared with averaged accuracy of the collection of unsupervised video summarization algorithms. Details on the length of original and summary videos are provided in Table 7.3.

As shown in Table 7.2, *GuideSum* achieves much higher accuracy than all other competing algorithms. Besides outperforming the group of unsupervised video summarization algorithms due to its capability of effectively utilizing supervised information, *GuideSum* also beats *MMED* by a significant margin. The reasons for this superiority over *MMED* lies in the complexity of YouTube videos, where the ground truth summary video contains segments corresponding to several different types of events. For example, in the wedding videos, the ground truth annotation of summary video contains events such as bride walking down the aisle, ring ceremony, couple's first dance, and cutting the wedding cake. The significant difference among those segments pose

Table 7.3: Length of both the original and summary videos (seconds) for 3 classes of YouTube videos used in experiments.

Video Class	Video	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Wedding	**Original**	2644	556	5852	1359	3001	1398	4275	3152	2452	2385
	Summary	74.9	63.2	131.0	67.5	81.2	58.4	109.3	75.8	61.0	64.9
FigureSkating	**Original**	527	522	560	479	515	382	403	504	496	570
	Summary	41.0	45.5	42.7	45.1	49.2	43.8	41.6	52.0	47.7	56.3
CarRacing	**Original**	2334	2143	1704	1404	1782	1604	1795	1619	1240	1356
	Summary	39.5	26.4	28.9	25.4	29.3	32.0	45.8	27.1	23.1	30.4

serious challenge for any event detection method, including *MMED*. Therefore, we can see that *GuideSum* is especially suited for videos depicting complicated events, such as wedding, parties, racing and various sports events.

Besides quantitative measures, we also show the summary videos automatically generated by *GuideSum* for the 3 scenarios. Specifically, Figure 7.4 shows frames taken from the summary video generated by *GuideSum* for a testing wedding video. From Figure 7.4, we can clearly see the following storyline: (1) bride gets ready in the dressing room; (2) bride walks down the aisle; (3) exchange of vows and the ring ceremony; (4) couple's first dance; (5) cutting the wedding cake and sharing a piece. Similarly, Figure 7.5 shows the automatically generated summary video for a figure skating video, where the summary video focuses on the exciting moments of the skating sequence, including (1) various spinning moves; (2) different types of jumping stunts; (3) the finishing move in the skating program. Finally, the summary video for car racing video is shown in Figure 7.6, which includes both the storyline and exciting moments of car racing: (1) standing start; (2) passing or being passed by other cars; (3) possibly running out of the circuit; (4) crossing the finish line. After the automatic summarization by *GuideSum*, a user could then focus only on a small portion of the usually very long racing videos, without missing the exciting moments.

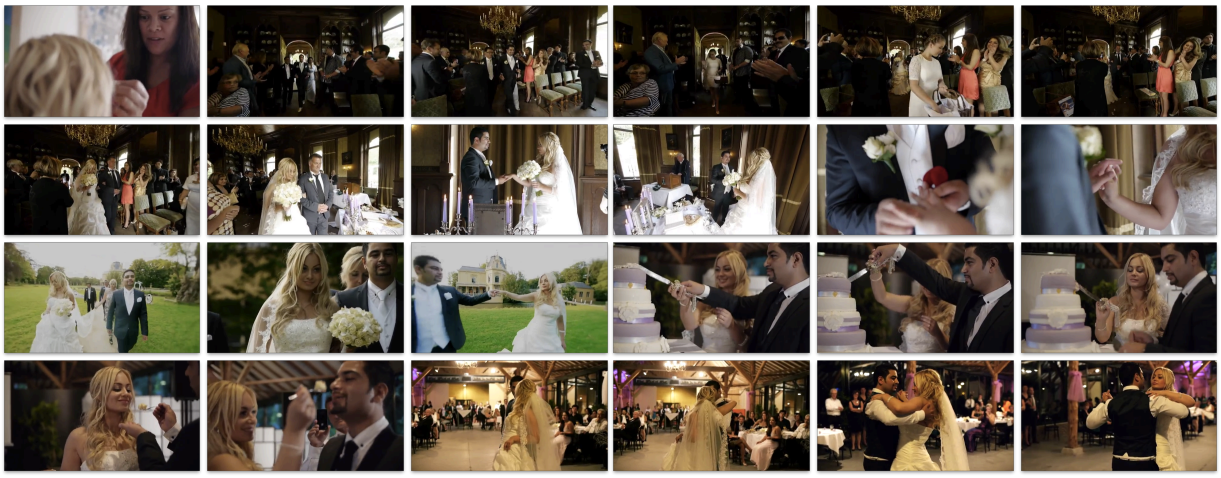

Figure 7.4: (Best viewed in color and zoom-in.) Some frames of the summary video generated by *GuideSum* for a wedding video (frames are organized from left to right, then top to bottom in temporal order).

Figure 7.5: (Best viewed in color and zoom-in.) Some frames of the summary video generated by *GuideSum* for a figure skating performance video (frames are organized from left to right, then top to bottom in temporal order).

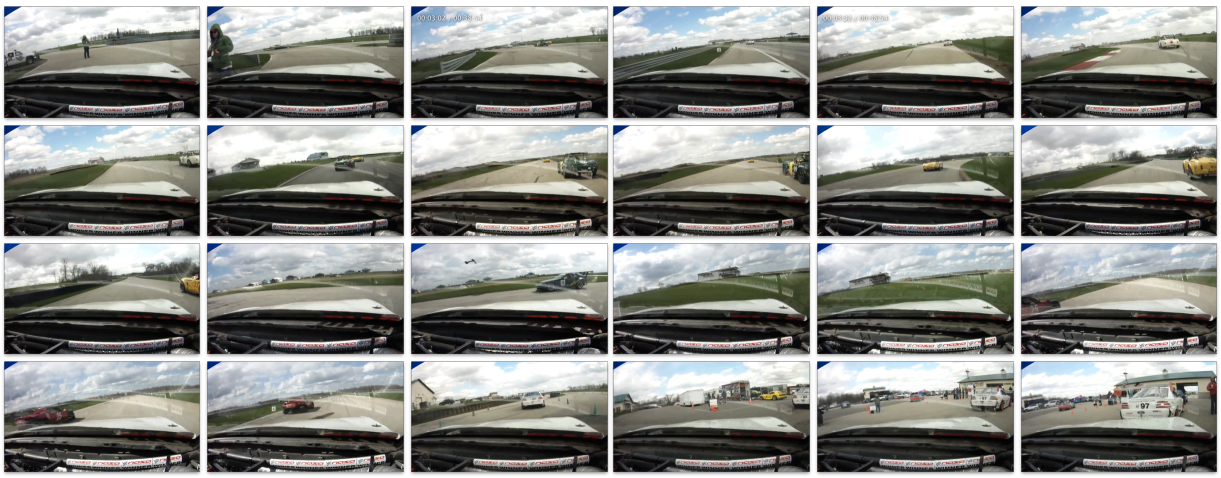

Figure 7.6: (Best viewed in color and zoom-in.) Some frames of the summary video generated by *GuideSum* for a car racing video recorded by car mounted camera (frames are organized from left to right, then top to bottom in temporal order).

7.5 Summary

GuideSum effectively utilizes user generated ground truth annotation for summary video, to significantly boost the accuracy of video summarization on videos of similar nature. Specifically, we formulate supervised video summarization as a max margin learning problem, where video segments selected into the ground truth summary video are treated as positive examples, and the remaining excluded video segments are used as negative examples. Moreover, *GuideSum* augments the set of positive examples by simulating partial events from the ground truth summary video, enabling correct identification of not only full sequence of events, but also partial sequences. Efficient optimization framework based on *CCCP*, *Cutting Plane* algorithm and *ADMM* is provided to solve the optimization problem in *GuideSum*. Finally, experimental results on both synthetic data and real world data, including surveillance videos and YouTube videos are provided to demonstrate the effectiveness of *GuideSum* in utilizing supervised information for accurate video summarization.

Part III

Conclusion

Chapter 8

Discussion

This dissertation presents a considerable step towards understanding large scale image collection and temporally long or even endless video sequences. We believe that with the ever increasing amount of digital contents generated by consumers and institutions, this direction of research becomes more important in both algorithmic development and real world application. In this chapter, we summarize the contributions and key observations of our work again, and discuss future research directions that go beyond our current achievement.

8.1 Key Observations and Contributions

As the concluding remarks of this thesis, we recapitulate the key observations and contributions.

Large Scale Image Classification

- Our work represents an initial foray to systematically utilizing information residing in hierarchical structure among image classes, for multi-way classification on super large-scale image data sets. First, the loss function used in our formulation weighs differentially for different misclassification outcomes: misclassifying an image to a category that is close to its true identity receives less penalty than misclassifying it to a totally unrelated one. Second, in an image classification problem with thousands of categories, it is not realistic to assume that all of the classes share the same set of relevant features. That is to say, a subset of highly related categories may share a common set of relevant features, whereas weakly related categories are less likely to be affected by the same features. Consequently, our method utilizes an overlapping-group-lasso penalty to achieve this type of structured sparsity at multiple levels of the hierarchy. Computationally, we propose a proximal gradient based method for solving the resulting non-smooth optimization problem, where the super large scale of the problem is tackled by map-reduce parallel computation.
- We then study even more challenging image classification problem, where the number of image classes in the collection could be tens of thousands or even more. Facing such challenge, our proposed *sparse output coding* method effectively breaks a multi-class classification problem with huge concept space into a two-step procedure: learning optimal coding matrix to assign codeword for each class in the original problem, followed by prob-

abilistic decoding to assign labels via *maximum a posteriori* criterion to testing images. Effectiveness of *sparse output coding* is demonstrated on large scale image categorization, with images from nearly 16 thousand classes. The fact that *sparse output coding* takes less bit predictors than *one-vs-rest* multi-class classification while achieving better accuracy, renders our proposed approach especially promising when scaling up to human cognition level multi-class classification.

Large Scale Video Understanding

- We propose a fully unsupervised dynamic sparse coding approach for detecting unusual events in videos based on online sparse re-constructibility of query signals from an atomically learned event dictionary, which forms a sparse coding bases. Based on an intuition that usual events in a video are more likely to be reconstructible from an event dictionary, whereas unusual events are not, our algorithm employs a principled convex optimization formulation that allows both a sparse reconstruction code, and an online dictionary to be jointly inferred and updated. Our algorithm is completely unsupervised, making no prior assumptions of what unusual events may look like and the settings of the cameras. The fact that the bases dictionary is updated in an online fashion as the algorithm observes more data, avoids any issues with concept drift. Experimental results on hours of real world surveillance video and several Youtube videos show that the proposed algorithm could reliably locate the unusual events in the video sequence, outperforming state-of-the-art methods.
- Based on our work on unusual event detection, we propose a principled way of generating short video highlight summarizing the most important and interesting contents of a potentially very long video, which is costly both time-wise and financially for manual processing. Specifically, our method learns a dictionary from given video using *group sparse coding*, and updates atoms in the dictionary on-the-fly. A highlight of the given video is then generated by combining segments that cannot be sparsely reconstructed using the learned dictionary. The online fashion of our proposed method enables it to process arbitrarily long videos and starts generating highlights before seeing the end of the video, both attractive characteristics for practical applications. Theoretical analysis of the proposed method, together with experimental results on hours of surveillance and YouTube videos are provided, demonstrating the effectiveness of our method.
- We propose a supervised video summarization algorithm to address the gap between the automated characterization of the important video segments and the human perception about the importance of a segment. Specifically, purely unsupervised approaches could potentially miss some of the key components, or incorporate unimportant events. On the other hand, in our proposed max margin video summarization approach, users are asked to generate summaries for a subset of a class of videos of similar nature. Given such manually generated summaries, our method learns the preferred storyline within the given class of videos, and automatically generates summaries for the rest of videos in the class, capturing the similar storyline as in those manually summarized videos. Our algorithm is capable of automatically selecting the optimal number of unique events in the summary video and understanding not only full but also partial events. Experimental results on a collection of

surveillance and YouTube videos demonstrate its effectiveness.

Although we have focused our discussion and study on visual data in this thesis, i.e., images and videos, our proposed algorithms could easily go beyond vision problems. For example, the sparse output coding framework could be readily applied to large scale text categorization, for learning classifiers capable of separating hundreds of thousands of text categories. Moreover, the unusual event detection algorithm could be applied to many other types of time series data, such as speech, financial market data, etc., for anomaly detection.

8.2 Future Directions

In this dissertation, we have proposed a set of algorithms for categorizing large collection of image data, and automatically detecting and summarizing the most salient and information-bearing portions of temporally long or even endless videos. In spite of their significant achievement for a wide range of novel and challenging problems, we believe much remains to be done along this line of research. Here, we propose several ideas for future projects to explore further the extension of our approach.

Learning mid-level image descriptors through latent space model

Besides training effective classifiers for separating tens of thousands of image classes, learning feature representations for images is also a crucial piece in successful understanding of large scale image collection. The first possible work along this line could focus on developing algorithms for learning mid-level image descriptors (features) based on corpus-wide visual-semantic knowledge and predictive latent-space representation.

It is widely accepted that designing proper features plays crucial role in several image understanding tasks, including recognition, identification, detection, etc. Most previous works employed low-level features, with examples such as SIFT, HOG, GIST, which capture the local characteristic of images. Then classifiers are learned based on those low-level features for inferring the high-level category labels. However, it is hardly sufficient to use only local gradient distribution (such as SIFT or HOG) for high-level image understanding. We believe it is crucial to bridge the gap between low-level features extracted from local patches of the original image and its categorical label. Specifically, we propose to use latent space model as means for learning mid-level representation of images. Based on such latent space model, a classifier will be learned. Consequently, we propose to develop algorithms for supervised latent space model and object-oriented dictionary learning from images, to provide basis for learning semantically meaningful mid-level image features. Specifically, current state-of-the-art image representation approaches only take into account the visual aspects of image data, but ignores all categorical information associated with training data. Consequently, image feature representation and classifier learning are formulated as two separated optimization frameworks. Our goal is to unify these two intrinsically closely correlated steps, and formulate them into a unified optimization framework, such that we could discover the optimal latent space model and dictionary for image representation, tailored towards accurate image classification.

One possible idea along this line is to learn mid-level image representations using sparse

topical coding [152], a non-probabilistic formulation of topic models for discovering latent representations of large collections of data. More precisely, sparse topical coding aims at learning both a dictionary and sparse representation of image features using the learned dictionary, where the dictionary terms will have the semantic meaning of an object, while the "word" corresponds to a feature of the object. Due to its non-probabilistic formulation, optimization over sparse topical coding will be efficient. Also, it will be intuitive and natural to integrate the sparse topical coding with supervised classifier learning, such as *sparse output coding* proposed in Chapter 4, in a unified framework, for simultaneous optimization over the latent feature representation and classifier learning.

Coupling recognition for different categories

Chapter 3 represents an initial foray into systematically utilizing the correlation among image classes for better classification. We propose to further explore the coexistence pattern across various objects, and couple the recognition of different categories of images. Specifically, related contents (e.g., a tent and a tribe man) are not predicted independently, but jointly through a model that leverages on label-structures such as WordNet, ImageNet, or other man-made ontologies. In this example, the recognition of a tent should give higher prior probability of a tribe man in the neighborhood of the tent, and vice versa. Consequently, the successful recognition rate of a tent and a tribe man should be higher than recognizing them separately.

Clearly, it is crucial to automatically extract those spatial coexistence patterns among various objects. To achieve such goal, based on our previous work of using sparse topical coding [147] as mid-level feature representation, we plan to define a Markov Random Field (MRF) over hidden topic assignment of super-pixels in an image to enforce the spatial coherence between neighboring regions, such that the MRF captures co-existing patterns among objects (such as tent and tribe man). Building an MRF on top of sparse topical coding gives the model ability to measure not only single object characteristic, but also inter-object correlation. Consequently, this model will assign higher probability to objects more likely to appear in close proximity, such as a tent and a tribe man, than less likely to co-exist objects such as a tent and a boat.

Cross-modal annotator

Images are sometimes accompanied by correlated text information, serving as the tag or annotation for such image, with the purpose of illustrating contents within the image. While we have mainly focused on mining information from images alone, it is also effective to combine the analyses of image and associated text in a unified subspace model. Specifically, instead of building latent space models for image and corresponding text separately, such as separate topic models for image and text, we could also build a unified subspace model for image and text, since they are closely correlated. The purpose of such unified model is to encourage the subspace model to leverage information from both image and text, such that the classification/detection task not only utilizes information from the visual data alone, but also from the textual data.

Specifically, we plan to start with the supervised multi-view restricted Boltzmann machine model, as an initial attempt to fuse information from both visual data and textual data. Upon successful investigation of the multi-view RBM model [25], we plan to dive deeper into models such as infinite SVM for multi-class classification, and infinite latent SVM for simultaneous

classification and feature selection [153, 154].

High-level feature representation for video summarization

Most interesting videos are centered around human activities. However, representing human actions in videos is hard because of the variation in visual appearance caused by changes in clothing, pose, articulation and occlusion. In this thesis, we use spatio-temporal interest points as feature representation for human activity. One potential problem with such representation is it totally ignores any structural information between those interest points. We argue that a better feature representation for video understanding is to define motion units that are tightly clustered both in configuration space (as might be parameterized by the locations of various joints), and in appearance space (as might be parameterized by pixel values in an image patch).

Supervised video summarization without original video

In Chapter 7, our formulation of supervised video summarization assumes the availability of not only the manually generated summary video, but also the original video, from which segments are selected to construct the summary. However, such original video might not always be readily available. For example, we can easily download professionally edited videos showing a person surfing in the ocean, but it could be difficult to obtain the original amateur footage based on which the edited video is generated. Given a new footage of surfing, one interesting task is to summarize it in a similar fashion as the professionally edited video. This task could be formulated as a classification problem where we are only given positive data points. Besides the lack of negative training data, the fact that the number of unique events in such collection of videos is unknown, also makes this supervised video summarization formulation interesting and challenging.

Cross event type video summarization

The video summarization algorithms proposed in Chapter 6 and Chapter 7 could be seen as the two extreme schemes of video summarization applications. Specifically, in Chapter 6, we assume no human intervention or guidance, and the algorithm uses motion and appearance information in the video to select unique video segments to be included into the summary. On the other hand, in Chapter 7, we formulate video summarization as a complicated event recognition problem, where the event types are intrinsically defined in the training videos and their annotations, and the algorithm aims at finding same events from the testing videos. We believe in real world video summarization applications, a framework between these two extreme cases might also be valuable. Specifically, given manually generated annotations as in Chapter 7, it is interesting to see how such supervised information could be effectively utilized in summarizing a different but related event type. For example, given wedding videos and manually generated annotations, cross event type video summarization could be used to summarize a collection of commencement videos. Clearly, wedding videos and commencement videos are different, with their own unique storylines. However, we can still observe the underlying shared characteristics of these two types of events. Specifically, in commencement videos, the video segments showing students walking towards the altar share similar characteristics with the bridal walking down the

aisle event in wedding videos. The reception of diploma might bear similar motion or appearance characteristics as the ring ceremony in wedding videos. Clearly, this is no longer a straight forward event recognition formulation, but a transfer of supervised information from one event type to another related but different event class.

8.3 Conclusion

In this dissertation, we identify the practical needs of machine learning based tools to unlock the huge collection of image and video data, generated by consumers and institutions. From these new challenges, we derive the thesis statement, which we would like to restate here as follows.

> **We aim to design machine learning algorithms to automatically analyze and understand large-scale image and video data.**

We categorize the required technologies into two research directions, which are (i) separating huge collection of images into large scale hierarchical class structure, (ii) automatic detection of salient and information-bearing portions from long videos, and generating summary / highlight videos with or without human guidance. All developed algorithms are aligned to accomplish the proposed research goal. We hope that this thesis can inspire others to pursue more interesting and practical projects at both algorithmic development and real world application towards understanding large scale image and video data.

Bibliography

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *PAMI*, 30:555 – 560, 2008. 2.2, 5.1, 5.2.3, 5.3.1, 5.3.1, 7.4.3

[2] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, A. Smola, and E. P. Xing. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics (AISTAT) 2011*. 1.1

[3] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *JMLR*, 1:113–141, 2001. 4.1, 4.1, 4.2.1, 4.1, 4.3, 4.3.3, 4.4.2

[4] A. Aner and J. Kender. Video summaries through mosaic-based shot and scene clustering. In *ECCV*, 2002. 2.2

[5] N. Babaguchi. Towards abstracting sports video by highlights. In *ICME*, 2000. 2.2

[6] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *JMLR*, 4:83–99, 2003. 1.1

[7] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *CVPR*, 2005. 2.1

[8] S. Bengio, F. Pereira, Y. Singer, and D. Strelow. Group sparse coding. In *NIPS*, 2009. 6.1

[9] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010. 2.1, I, 3, 4.2.1, 4.2.1, 4.4.1, 4.4.2, 4.4.3, 4.4.3, 4.5

[10] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012. 2.1

[11] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, 2009. 2.1

[12] A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *ALT*, 2009. 2.1, 4.4.2

[13] A. Binder, K.-R. Mller, and M. Kawanabe. On taxonomies for multi-class image categorization. *IJCV*, pages 1–21, 2011. 1.1

[14] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. 3.4.3

[15] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005. 5.2.1

[16] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. 1.1

[17] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *ICCV*, 2005. 2.2, 5.1, 5.2.3

[18] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 1.1

[19] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010. 4.4.2, 7.3.3

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011. 4.2.2, 4.2.2, 4.2.2, 2, 6.2.1, 6.2.1

[21] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32:13–47, March 2006. 3.2.2

[22] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, 2004. 2.1, 4.2.1

[23] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. 2.1

[24] Y. Caspi, A. Axelrod, Y. Matsushita, and A. Gamliel. Dynamic stills and clip trailers. *Vis. Comput.*, 22(9):642–652, 2006. 2.2

[25] N. Chen, J. Zhu, and E. Xing. Predictive Subspace Learning for Multi-view Data: a Large Margin Approach. In *NIPS*, 2010. 8.2

[26] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33 – 61, 1998. 5.2.2

[27] X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. P. Xing. Smoothing proximal gradient method for general structured sparse learning. In *UAI*, 2011. 3.1, 3.3, 3.3.1, 3.3.1

[28] P. M. Cheung and J. T. Kowk. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006. 7.3.1, 7.3.1

[29] C. Chu, S. Kim, Y. Lin, Y. Yu, G., A. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS*. 2007. 3.1, 3.3, 3.3.2

[30] Y. Cong, J. Yuan, and J. Luo. Towards scalable summarization of consumer videos via sparse dictionary selection. *TMM*, 14(1):66–75, 2012. (document), 6.2.3, 6.4.1, 6.2, 6.4.1, 7.4.3

[31] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 2:265–292, 2002. 4.1, 4.2.1

[32] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990. 1.1

[33] O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML*, 2004. 2.1, 4.4.2

[34] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *ACM MM*, 1998. 2.2

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1.1, 4.2.1, 4.4

[36] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 3.2.1

[37] J. Deng, A. Berg, and L. Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *CVPR*, 2011. 2.1

[38] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011. 2.1, I, 3

[39] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *JAIR*, 2:263–286, 1995. 4.1, 4.3

[40] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005. (document), 5.2.1, 5.2, 6.2.1, 7.4.1

[41] F. Dufaux. Key frame selection to represent a video. In *ICIP*, 2000. 2.2

[42] J. Eckstein and D. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1): 293–318, 1992. 4.2.2

[43] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2), 2004. 5.2.2

[44] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *CVPR*, 2012. 2.2, 6, 6.2.3

[45] S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *PAMI*, 32(1):120–134, 2010. 4.1, 4.1, 4.3

[46] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2.1

[47] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *PAMI*, 28: 594–611, 2006. 2.1

[48] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic label sharing for learning with many categories. In *ECCV*, ECCV'10, 2010. 1.1, 2.1

[49] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2 (1):17–40, 1976. 4.2.2

[50] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011. 2.1, I, 3, 4.4.2

[51] T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In *ICML*, 2011. 4.1, 4.2.1

[52] D. Goldman, B. Curless, D. Salesin, and S. Seitz. Schematic storyboarding for video visualization and editing. In *SIGGRAPH*, 2006. 2.2

[53] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *PAMI*, 29(12):2247–2253, 2007. 2.2, 6.2.4, 7.4.2

[54] A. Basharatand A. Gritai and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In *CVPR*, 2008. 2.2, 5.1

[55] R. Hamid, A. Johnson, S. Batta, A. Bobick, C. Isbell, and G. Coleman. Detection and explanation of anomalous activities: Representing activities as bags of event n-grams. In *CVPR*, 2005. 2.2, 5.1

[56] A. Hanjalic and H. Zhang. An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE TSCVT*, 9:1280–1289, 1999. 2.2

[57] D. Haussler. Convolution kernels on discrete structures. *Technical report*, 1999. 4.2.1

[58] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 107(2):191–202, 2014. 2.2, 7.2.2, 7.4.3

[59] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009. 2.1

[60] W. Hu, X. Xiao, Z. Fu, and D. Xie. A system for learning statistical motion patterns. *PAMI*, 28:1450 – 1464, 2006. 2.2

[61] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *NIPS*, 2008. 1.1

[62] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, 2010. 3.1, 3.3, 3.3.1

[63] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999. 7.3.1

[64] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:703–712, 1960. 7.3.2

[65] J. Kim and K. Grauman. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In *CVPR*, 2009. (document), 2.2, 5.1, 5.2.3, 5.3.1, 5.3.1, 5.1, 5.3.1, 5.3.1, 5.3.1, 2, 5.2

[66] S. Kim and E. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010. 3.2.3

[67] D. Koller and M. Sahami. Hierarchically classifying docuemnts using very few words. In *ICML*, 1997. 2.1

[68] A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham. The ecir 2010 large scale hierarchical classification workshop. *SIGIR Forum*, 44(1):23–32, 2010. 1.1, 4.4.2

[69] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009. 2.1

[70] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 2.1

[71] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 2.2

[72] K. Lange, D.R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Jounal of Computational and Graphical Statistics*, 9:1–59, 2000. 7.3.1

[73] I. Laptev. On space-time interest points. *IJCV*, 64:107 – 123, 2005. 5.2.1

[74] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 3.4.1

[75] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 2.1, 4.4.2, 4.4.3

[76] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. 2.1

[77] H. Lee, A. Battle, R. Rajat, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007. 1.1, 5.1, 5.2.2, 5.2.2

[78] Y. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, 2012. 2.2, 6

[79] B. Li and I. Sezan. Event detection and summarization in american football broadcast video. In *SPIE Storage ad Retrieval for Media Databases*, 2002. 2.2

[80] J. Li, S. Gong, and T. Xiang. Global behaviour inference using probabilistic latent semantic analysis. In *BMVC*, 2008. 2.2, 5.1

[81] L. Li, H. Su, E. Xing, and L. Fei-Fei. Object bank: A highlevel image representation for scene classification and semantic feature sparsification. In *NIPS*, 2010. 2.1

[82] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and svm training. In *CVPR*, 2011. 1.1, 2.1, 4.4.1, 4.4.2, 4.4.3

[83] D. Liu, G. Hua, , and T. Chen. A hierarchical visual model for video object summarization. *PAMI*, 2009. 2.2

[84] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 3.4.1, 4.4.1

[85] J. Luo, C. Papin, and K. Costello. Towards extracting semantically meaningful key frames from personal video clips: from humans to computers. *TCSVT*, 19(2):289–301, 2009. 6.4.1, 7.4.3

[86] S. P. Luttrell. Partitioned mixture distributions: An adaptive bayesian network for low-level image processing. In *IEEE Proceedings on Vision, Image and Signal Processing*, volume 141, pages 251–260, 1994. 7.3.1

[87] Z. Ma, Y. Yang, Z. Xu, S. Yan, N. Sebe, and A. Hauptmann. Complex event detection via multi-source video attributes. In *CVPR*, 2013. 2.2

[88] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009. 5.2.3

[89] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010. 6.2.2

[90] E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. In *CVPR*, 2000. 2.1

[91] T. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997. 1.1

[92] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $o(\frac{1}{k^2})$. *Doklady AN SSSR (translated as Soviet. Math. Docl.)*, 269:543–547, 1983. 3.3, 3.3.2

[93] N. Nilsson. *Learning Machines*. McGraw-Hill, 1965. 4.1, 4.3

[94] J. Oh, Q. Wen, J. lee, and S. Hwang. Video abstraction. *Video Data Mangement and Information Retrieval*, pages 321–346, 2004. 2.2

[95] S. Oh, J. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *IJCV*, 77(1–3):103–124, 2008. 2.2

[96] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, 2006. 2.1

[97] M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. 2.1

[98] M. Parsana, S. Bhattacharya, C. Bhattacharyya, and K. Ramakrishnan. Kernels on attributed pointsets with applications. In *NIPS*, 2007. 4.2.1

[99] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE TNN*, 15(1):45–54, 2004. 4.1, 4.3

[100] A. Patron-Perez, M. Marszalek, A. Zisserman, and I. Reid. High five: Recognising human interactions in tv shows. In *BMVC*, 2010. 2.2

[101] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, 2012. 2.1

[102] S. Perkins and J. Theiler. Online feature selection using grafting. In *ICML*, 2003. 5.2.2

[103] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 2.1

[104] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, 1996. 2.2

[105] B. Póczos, L. Xiong, and J. Schneider. Nonparametric divergence estimation with applications to machine learning on distributions. In *UAI*, 2011. 4.2.1

[106] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *ICCV*, 2007. 2.2

[107] O. Pujol, P. Radeva, and J. Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *PAMI*, 25(6):1001–1007, 2006. 4.1, 4.3

[108] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008. 2.1

[109] Z. Rasheed and M. Shah. Detection and representation of scenes in videos. *TMM*, 7(6): 1097–1105, 2005. 6.4.1, 7.4.3

[110] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012. 2.1

[111] A. Rav-Acha, Y. Pritch, and S. Peleg. Making a long video short. In *CVPR*, 2006. 2.2

[112] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004. 1.1, 4.4.2, 4.4.3

[113] R. Salakhutdinov, A. Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011. I, 3

[114] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010. 2.2

[115] R. Schapire. Using output codes to boost multiclass learing problems. In *ICML*, 1997. 4.1

[116] R. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. Adaptive Computation and Machine Learning Series. Mit Press, 2012. 4.2.1

[117] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, 2008. 2.2

[118] M. Smith and T. Kanade. Video skimming and characterization through the combination of image and language understanding. In *CVPR*, 1997. 2.2

[119] A. J. Smola, S.V.N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *AISTATS*, 2005. 7.3.1, 7.3.1, 6

[120] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *PAMI*, 22:747 – 757, 2000. 2.2

[121] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Learning hierarchical models of scenes, objects, and parts. In *CVPR*, 2005. 2.1

[122] C. Taskiran, A. Amir, D. Ponceleon, and E. Delp. Automated video summarization using speech transcripts. In *SPIE*, 2002. 2.2

[123] J. Tenenbaum and W. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12:1247–1283, 2000. 2.1

[124] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, pages 91–108, 2005. 3.3

[125] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999. 1.1

[126] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004. 2.1

[127] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 30:1958–1970, 2008. 1.1

[128] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010. 2.1

[129] B. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), 2007. 2.2

[130] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–484, 2005. 7.3.2

[131] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr using stochastic intersection kernel machines. In *ICCV*, 2009. 2.1

[132] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 3.4.1

[133] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception by hierarchical bayesian models. In *CVPR*, 2007. 2.2, 5.1

[134] K. Weinberger and O. Chapelle. Large margin taxonomy embedding for document categorization. In *NIPS*, 2008. 2.1

[135] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, pages 1–38, 2012. 4.2.2, 4.2.2, 4.2.2, 4.2.2, 2

[136] J. Weston, S. Bengio, and N. Usunier. Wsabie: scaling up to large vocabulary image annotation. In *IJCAI*, 2011. (document), 2.1, 4.4.1, 4.4.3, 4.4.3, 4.5

[137] W. Wolf. Key frame selection by motion analysis. In *ICASSP*, 1996. 2.2

[138] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 4.4, 4.4.1

[139] Eric P. Xing. Topic models, latent space models, sparse coding, and all that: A systematic understanding of probabilistic semantic extraction in large corpus. In *ACL (Tutorial Abstracts)*, 2012. 1.1

[140] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15: 915–936, 2003. 7.3.1

[141] H. Zhang. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, 1997. 2.2, 6

[142] X. Zhang, L. Liang, and H. Shum. Spectral error correcting output codes for efficient multiclass recognition. In *ICCV*, 2009. 4.4.2

[143] Y. Zhang and J. Schneider. Maximum margin output coding. In *ICML*, 2012. 2.1

[144] B. Zhao and E. Xing. Sparse output coding for large-scale visual recognition. In *CVPR*, 2013. 1.1, 1.2

[145] B. Zhao and E. Xing. Quasi real-time summarization for consumer videos. In *CVPR*, 2014. 1.1, 1.2, 7.4.3

[146] B. Zhao and E. Xing. Hierarchical feature hashing for fast dimensionality reduction. In *CVPR*, 2014. 1.2

[147] B. Zhao, L. Fei-Fei, and E. Xing. Image segmentation with topic random fields. In *ECCV*, 2010. 1.2, 8.2

[148] B. Zhao, L. Fei-Fei, and E. Xing. Online detection of unusual events in videos via dynamic

sparse coding. In *CVPR*, 2011. 1.1, 1.2

[149] B. Zhao, L. Fei-Fei, and E. Xing. Large-scale category structure aware image categorization. In *NIPS*, 2011. 1.1, 1.2

[150] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR*, 2004. 2.2, 5.1, 5.3.1

[151] D. Zhou, L. Xiao, and M. Wu. Hierarchical classification via orthogonal transfer. In *ICML*, 2011. 2.1, I, 3, 4.2.1

[152] J. Zhu and E.P. Xing. Sparse topical coding. *arXiv preprint arXiv:1202.3778*, 2012. 8.2

[153] J. Zhu, N. Chen, and E. Xing. Infinite svm: a dirichlet process mixture of large-margin kernel machines. In *ICML*, 2011. 8.2

[154] J. Zhu, N. Chen, and E. Xing. Infinite latent svm for classification and multi-task learning. In *NIPS*, 2011. 8.2

[155] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003. 4.3.1, 4.3.2, 4.3.2

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
www.ml.cmu.edu