THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

UNDERSTAND VIDEO EVENT BY EXPLOITING SEMANTIC AND TEMPORAL INFORMATION FOR CLASSIFICATION AND RETRIEVAL

Chao Li

Master of Engineering

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2018

School of Information Technology & Electrical Engineering

Abstract

Thanks to the rapid advances in mobile video capturing devices and in network connections, more and more users prefer to use videos to record their daily life. Video is becoming a common means of recording everything from marriage proposals to how to repair an appliance. The number of user generated videos that record complex events is exploding on the Web. Therefore, technologies to assist in automatically understanding video events are in high demand to analyse and manage this exploding amount of video content.

The main challenges of video event understanding come from the diversity and complexity of the video content and its temporal nature. To deal with these challenges, in this thesis, we exploit semantic and temporal information for video event classification and retrieval, which are two main tasks about video event understanding. Our work is organized into four main chapters.

For video event classification, in Chapter 3, to address the diversity and complexity of the video content, we define two types of latent concepts, i.e. a static-visual concept at frame-level and an activity concept at segment-level, to alleviate the influence of high intra-class variation. Furthermore, we propose a data-driven hierarchical structure of latent variables to discover the latent concepts, where temporal information is utilized in the discovery process. In Chapter 4, Long Short-term Memory (LSTM) is employed to capture the temporal information in videos. A novel temporal attention model is proposed, which enables our framework to focus on the most related shots during the classification procedure. Moreover, weak semantic relevance is incorporated as fine-grained guidance (at shot-level) for the proposed temporal attention model to further enhance the classification performance. In contrast to the work in Chapter 3, where the underlying semantic information is organized as latent concepts and the latent concept discovery process is data-driven, in the proposed framework in Chapter 4, semantic information is formalized as weak semantic relevance and is employed as explicit supervision.

Recently, hashing has been evidenced as an efficient and effective method to facilitate large-scale video retrieval. Most of existing hashing methods are based on static features. The intrinsic temporal patterns embedded in videos have also shown their discriminative power for similarity search. However, how to leverage the strengths of both these aspects remains unknown. In Chapter 5, we propose to jointly model two essential aspects of videos (i.e. temporal pattern and static feature), with two encoders, for unsupervised video event retrieval. For jointly modelling, three learning criteria for generating high-quality hash codes are imposed on the two encoders. To further explore how to utilize features in regard to both aspects more effectively, in Chapter 6, we propose a novel information filtering mechanism which we call Adaptive Selection, which exploits the complementary advantages of the two aspects for supervised video event retrieval.

Declaration by Author

Publications during candidature

Conference papers:

- **Chao Li**, Jiewei Cao, Zi Huang, Lei Zhu and Heng Tao Shen. Leveraging Weak Semantic Relevance for Complex Video Event Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3647–3656, 2017.
- **Chao Li**, Yang Yang, Jiewei Cao and Zi Huang. Jointly Modeling Static Visual Appearance and Temporal Pattern for Unsupervised Video Hashing. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 9–17, 2017.
- Xiaoshuai Sun, Jiewei Cao, **Chao Li**, Lei Zhu and Heng Tao Shen. Web-based Semantic Fragment Discovery for On-line Lingual-Visual Similarity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 182–188, 2017.
- Jiewei Cao, Zi Huang, Peng Wang, **Chao Li**, Xiaoshuai Sun and Heng Tao Shen. Quartet-net Learning for Visual Instance Retrieval. In *Proceedings of the ACM Conference on Multimedia*, pages 456–460, 2016.

Journal papers:

- **Chao Li**, Zi Huang, Yang Yang, Jiewei Cao, Xiaoshui Sun and Heng Tao Shen. Hierarchical Latent Concept Discovery for Video Event Detection. *IEEE Transactions on Image Processing*, pages 2149–2162, 2017.

Publications included in this thesis

Chao Li, Zi Huang, Yang Yang, Jiewei Cao, Xiaoshui Sun and Heng Tao Shen. Hierarchical Latent Concept Discovery for Video Event Detection. *IEEE Transactions on Image Processing*, pages 2149–2162, 2017. Incorporated as Chapter 3.

Contributor	Statement of contribution
Chao Li (Candidate)	Conception and design (75%) Analysis and interpretation (75%) Drafting and production (75 %)
Zi Huang	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)
Yang Yang	Conception and design 5%) Analysis and interpretation (5%) Drafting and production (5 %)
Jiewei Cao	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)
Xiaoshui Sun	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)
Heng Tao Shen	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)

Chao Li, Jiewei Cao, Zi Huang, Lei Zhu and Heng Tao Shen. Leveraging Weak Semantic Relevance for Complex Video Event Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3647–3656, 2017. Incorporated as Chapter 4.

Contributor	Statement of contribution
Chao Li (Candidate)	Conception and design (80%) Analysis and interpretation (75%) Drafting and production (75 %)
Jiewei Cao	Conception and design (5%) Analysis and interpretation (10%) Drafting and production (5 %)
Zi Huang	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (10 %)
Lei Zhu	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)
Heng Tao Shen	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (5 %)

Chao Li, Yang Yang, Jiewei Cao and Zi Huang. Jointly Modeling Static Visual Appearance and Temporal Pattern for Unsupervised Video Hashing. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 9–17, 2017. Incorporated as Chapter 5.

Contributor	Statement of contribution
Chao Li (Candidate)	Conception and design (80%) Analysis and interpretation (80%) Drafting and production (80 %)
Yang Yang	Conception and design (5%) Analysis and interpretation (10%) Drafting and production (5 %)
Jiewei Cao	Conception and design (10%) Analysis and interpretation (5%) Drafting and production (5 %)
Zi Huang	Conception and design (5%) Analysis and interpretation (5%) Drafting and production (10 %)

Contributions by others to the thesis

No contributions by others.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Research Involving Human or Animal Subjects

No animal or human subjects were involved in this research.

Acknowledgments

Thanks to my advisor Dr Zi Huang, for her invaluable advice during my research and also for her help in daily life as one of my best friends.

In addition, I really enjoyed working with my collaborators Prof. Heng Tao Shen, Prof. Yang Yang, Prof. Fumin Shen and Mr Jiewei Cao *et al.*

Furthermore, I will always appreciate the friendships with all my colleagues in the DKE group. They will be a precious memory in my life.

Thanks also to my parents and sister for their full support and understanding, which enabled me to dedicate myself to completing my Ph.D. without distractions, without worries.

Financial support

This research was supported by the following scholarships:

- CSC – China School Council
- UQRTTUIT – UQ Research Training Tuition Fee Offset
- UQITUITON – UQ International Tuition
- CDA-2 – Candidate Development Award

Keywords

video event understanding, temporal information, deep learning, computer vision.

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080104, Computer Vision, 60%

ANZSRC code: 080109, Pattern Recognition and Data Mining, 40%

Fields of Research (FoR) Classification

FoR code: 0801, Artificial Intelligence and Image Processing, 100%

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid advancements in technology in mobile video capturing devices and the availability of high-speed mobile networks, more and more users prefer to use videos to record their daily life rather than photos. Video is becoming a common means of recording various events from *marriage proposals* to *how to repair an appliance*. In the meanwhile, an increasing number of social media Web services, such as YouTube, Snapchat, and Twitch, provide great platforms for video sharing. The number of user generated videos that record complex events [101] (e.g. “a birthday party”, “a wedding ceremony” or “repairing a vehicle”) is exploding on the Web. In this context, advanced techniques to understand video events are in high demand. Due to the complexity and diversity of video content, video event understanding remains challenging. In this thesis, we focus on leveraging the semantic and temporal information embedded in videos to capture high-level concepts, which are the main components of an event. In detail, semantic information is abstracted as high-level concepts or semantic relevance to alleviate the diversity and complexity of the video content. Temporal information is carefully modelled to assist the above procedure. We have examined our methods for video event classification and video event retrieval to demonstrate their performance of in understanding events.

1.1 Video Event Classification

Video event classification is an important task in the computer vision and multimedia communities [41, 101, 128], where classifiers are trained to classify event videos into different categories. An

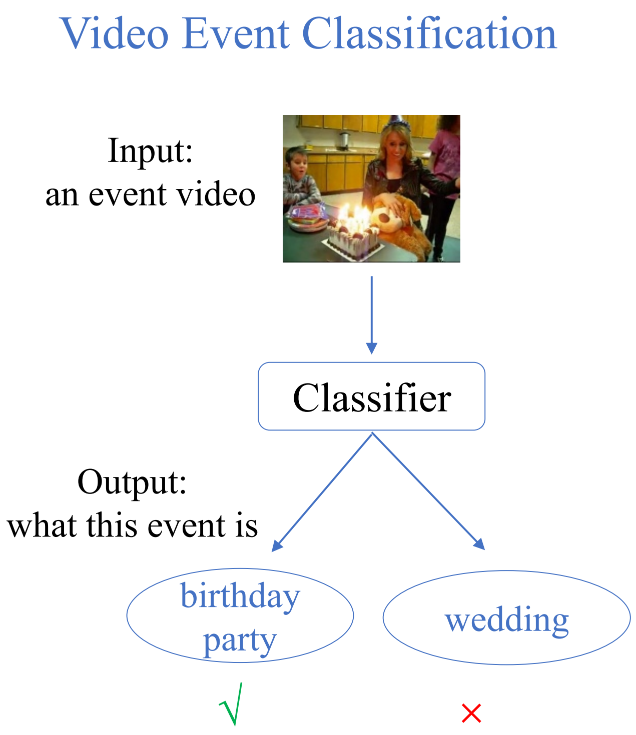

FIGURE 1.1: Illustration of the video event classification task. Input an event video, the event classifier will categorize it automatically.

illustration of video event classification task is shown in Figure 1.1. The main challenges of video event classification are in regard to the following aspects. Firstly, most complex events (e.g. “wedding ceremonies” or “birthday parties”) have unconstrained content, which typically involves not only various entities such as objects, people and animals, etc., but also diverse interactions between these entities (e.g. “a bride holding flowers” and “cutting a cake with knife”). Secondly, videos of the same type of event may appear very different visually. For example, a birthday party can be held in a backyard covered with lawn or in a dining hall with tables. The diversity of unconstrained content and the visual variety within an individual event category is considered to be the intra-class variation. Due to the temporal nature of videos, it is more difficult to understand videos as compared to static images. For example, in a video that records “parking a vehicle”, we can typically see the following sequence of activities: “vehicle entering parking space”, “engine turning off” and “driver getting off vehicle”. Once we have watched the complete sequence, we can easily tell that this is a video recording “parking a vehicle”. It is the embedded temporal information that makes this video an event of "parking a vehicle". If there are only a set of images of "driver is touching vehicle door", "engine is running" and "vehicle is near parking space" available, we may misunderstand this event as "vehicle departing". Hence, the plentiful underlying temporal information in videos needs to be thoroughly utilized to achieve a high-quality classification result.

1.1.1 Hierarchical Latent Concept Discovery

Semantic information is generally defined as concepts that can be used to describe an object, scene or activity etc. It is important for video event classification. High-level concepts abstracted from semantic information can be used to alleviate the intra-class variation caused by the diversity and complexity of the video content. For example, in videos recording event "reversing a vehicle", we can use the concept "vehicle" to describe all vehicles regardless of what colour, size or model they are (e.g. "black car", "silver SUV" or "small tractor"). Although these differences lead to high intra-class variation in low-level visual features, they can be abstracted into higher-level concepts (e.g. "vehicle"), which are easier to handle by an event classifier. How to automatically discover, model and utilize semantic information to facilitate video event classification remains an open problem.

In this thesis, a novel hierarchical classification model is proposed to deliberately unify the processes of underlying semantics discovery and event modelling. Specifically, in contrast to most approaches based on manually pre-defined concepts, we devise an effective model to automatically discover video semantics by hierarchically capturing latent static-visual concepts at the frame-level and latent activity concepts (i.e. the temporal sequence of static-visual concepts) at the segment-level. Temporal information is utilized in the discovery process. The unified model not only enables a discriminative and descriptive representation for videos, but also alleviates the error propagated from video representation to event modelling, which exists in previous methods. A max-margin framework is employed to learn the model. Extensive experiments on four challenging video event datasets, i.e. MED11, CCV, UQE50 and FCVID, are conducted to verify the effectiveness of the proposed method. The detailed techniques are discussed in Chapter 3.

1.1.2 Weak Semantic Relevance Utilization

Apart from discovering the underlying semantic information embedded in video content as latent concepts, we have also performed studies on formalizing semantic information acquired from the Web as weak semantic relevance, which is employed as explicit fine-grained supervision.

Conventionally, human annotations are widely used for training video event classifiers, a practice which is labour-consuming. Another option is to use weak semantic annotations, which can be harvested from Web-knowledge (i.e. knowledge acquired from the Web, such as from Wikipedia, Flicker and Google Images), without involving any human interaction. For example, image annotators, which are trained on labeled Web images, could be used to annotate video frames to generate semantic annotations. The quality of the semantic annotations generated in this way cannot be guaranteed to be as reliable as human-generated annotations. Thus, we call them weak semantic annotations and it is infeasible for them to be directly utilized for event classification without considering their reliability. In Chapter 4, we propose a new approach to automatically maximize the utility of weak semantic annotations (formalized as the semantic relevance of video shots to the target event) to facilitate video event classification. A novel attention model is designed to determine the attention score of each video shot, where the weak semantic relevance is considered as attentional guidance. In detail, our model jointly optimizes two objectives at different levels. The first one is the classification loss corresponding to video-level groundtruth labels, and the second is the shot-level relevance loss corresponding to weak semantic relevance. A long short-term memory (LSTM) layer is used to capture the temporal information that exists in the shots of a video. In each timestep, the LSTM employs the attention model to weigh the current shot under the guidance of its weak semantic relevance to the event of interest. Thus, it can automatically exploit weak semantic relevance to assist in video event classification. We conduct experiments on three large-scale benchmark video datasets and the experimental results demonstrate the superior performance of the proposed method.

1.2 Video Event Retrieval

Content-based visual retrieval has attracted wide attention from information retrieval, multimedia, and database communities [32,70,116–118], especially with the exploding multimedia content on the

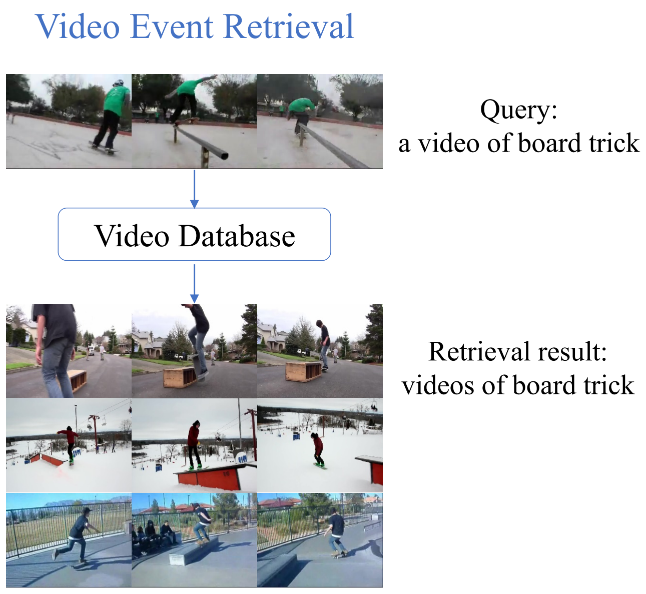

FIGURE 1.2: Illustration of the video event retrieval task. Given an event video as a query, the retrieval system will retrieve a list of videos which are related to the query video from a video database.

Web in recent years. An demonstration of video event retrieval task is shown in Figure 1.2. Hashing-based methods for this task have also caught the interest of researchers because of their advantages in reducing computational and storage cost. Hashing-based methods transform visual content into a small set of binary codes, which significantly potentially enhances the retrieval efficiency in the Hamming space. Moreover, the storage cost is greatly reduced by using binary codes instead of real-value codes.

Typically, hashing methods can be divided into two main categories: data-independent methods and data-dependent methods. For data-independent methods [6,24,82], the hash functions are defined independently of the dataset and the data distribution information is neglected. Thus, some useful information carried by the data, e.g. semantic labels and pair-wise similarities of samples, is not utilized. To take such information into consideration, many data-dependent methods have been proposed, such as spectral hashing [110], principal component analysis based hashing [25, 64], anchor graph-based hashing [68], etc. They are also referred to as learning to hash methods, whose hash functions are learned from the provided training datasets. It has been proved that in many cases, short binary codes generated by data-dependent methods can achieve comparable or even better performance than the longer binary codes generated by data-independent methods. Thus, in this thesis, we focus on the learning to hash methods.

In the past decade, hashing methods for content-based image retrieval have been extensively studied [15, 66, 87, 91, 107, 117]. On the contrary, video hashing has received limited attention from the community. In contrast to images, videos not only contain diverse and complex visual information in each frame, but also carry certain temporal information, such as short-term actions and long-term events, across frames [57, 58, 126]. A video is actually beyond a simple set of images. In addition to the static visual appearance presented in video frames, the temporal information embedded in the sequence of video shots provides valuable supplementary information that can be used for effective searches. How to take both static and dynamic (i.e. temporal) information into account in video hashing needs to be investigated comprehensively.

1.2.1 Static Feature and Temporal Pattern Modelling

Most of video hashing methods generate hash codes solely based on static visual features, which are expected to capture the appearance of videos. The intrinsic temporal pattern embedded in videos has also shown its discriminative power for similarity search, and is explored and utilised in some recent studies. However, how to leverage the strengths of both aspects has not yet been determined.

We propose a pioneering framework to jointly model static visual features and temporal patterns for video hash code generation, as both of these are believed to carry important information to generate an effective hash function. A novel unsupervised video hashing framework is also designed with a hash function comprised of two encoders, the temporal encoder and the appearance encoder. The two encoders are self-supervised and are designed to be able to reconstruct the temporal pattern of videos and the visual feature of frames, respectively. Three learning criteria are imposed to jointly learn the two encoders: minimal binarization loss, balanced hash codes and independent hash codes. From the extensive experiments conducted on two large-scale video datasets (i.e. FCVID and ActivityNet), we have confirmed the superior performance of our method compared to the state-of-the-art video hashing methods. Detailed explanations of the work are given in Chapter 5.

1.2.2 Video Retrieval via Adaptive Selection

Deep learning [3, 55] is a kind of machine learning method based on learning data representations. It achieves state-of-the-art performance on many tasks in computer vision literature, such as image classification [29, 47, 88], language modelling [12, 16] and captioning [104, 114, 124]. By leveraging deep learning techniques, deep learning to hash methods [8, 50, 60, 65, 66, 113] have demonstrated promising performance on large-scale image retrieval. Conventionally, deep learning to hash methods employ convolutional neural networks (CNNs) to learn real-valued features from original RGB images and feed them to hash functions to generate binary codes. In the training phase, the features and hash functions are simultaneously learned.

However, deep learning to hash methods for video retrieval have not been thoroughly studied. As discussed in the previous section, it has proven that temporal patterns and static visual features are both important for understanding video. How to represent videos by considering both aspects in an effective way to generate high quality hash codes is essential for the learning to hash method. Therefore, new form of video feature representation that takes both aspects into account is required rather than a straightforward concatenation.

We propose a dual-stream deep network to adaptively model both static visual features and temporal patterns for video hashing. A novel Adaptive Selection (AS) mechanism is designed to adaptively select useful components from the original input with respect to each of the above two aspects. The AS takes into account temporal patterns for static visual feature selection, and vice versa, where the complementary advantages of static visual features and temporal patterns are well exploited. An intermediate video representation is generated in the form of an optimal integration of static visual features and temporal patterns, based on which hash codes are finally produced. Comprehensive performance studies have been conducted, which verify the promising performance of the proposed dual-stream framework compared to the state-of-the-art video hashing methods. The detailed techniques are discussed in Chapter 6.

1.3 Benchmark Datasets for Video Event Understanding

In this thesis, we conduct experiments on a range of standard benchmark datasets to evaluate our methods. We briefly introduce them as follows:

1. **MED11** [101] contains 2047 diverse videos collected from the internet. These videos fall into 15 events.

2. **MEDTest14** [100] is a commonly-used benchmark dataset covering 20 events for complex video event classification. Each event has 100 positive training examples, and all events share about 5,000 negative training examples. The test set has approximately 23,000 videos.

3. **CCV.** This dataset contains 9,317 YouTube videos covering 20 event categories. The event names and train/test splits can be found in the original paper [44].

4. **FCVID.** To the best of our knowledge, FCVID [43] is one of the largest video datasets currently available for event classification. It consists of 91,223 Web videos annotated manually into 239 categories. The total duration of all the videos is 4,232 hours and the average video duration is 167 seconds.

5. **UQE50.** Video dataset UQE50 [58] (UQ Event dataset with 50 pre-defined events) contains 3,462 event videos divided into different event categories and 18,495 distractors that are irrelevant to any pre-defined events. All videos in this dataset are downloaded from YouTube. The videos from UQE50 are all of hot global events that occurred in the last few years and they contain far more complex patterns than other datasets that mainly contain activity or action sequences.

6. **ActivityNet** [30] was recently released for complex human activity recognition. It comprises 28K of videos of 203 activity categories collected from YouTube. The video durations range from several minutes to half an hour and the total length of the whole dataset is 849 hours. Many of the videos in this dataset are shot by amateurs in uncontrolled environments, where the variances within the same activity category are often large.

To achieve fair comparisons and keep consistent with other state-of-the-art methods, we adopt particular experimental protocols on these datasets. More details are provided in the experiment section in each main chapter.

1.4 Thesis Overview

The thesis is organized as follows: In Chapter 2, we review the related work on video event understanding. In Chapter 3, we propose a data-driven hierarchical structure of latent variables to discover latent concepts for event classification. In Chapter 4, we incorporate weak semantic relevance, as fine-grained guidance (at shot-level) to the proposed temporal attention model, to facilitate video event classification. In Chapter 5, we jointly model two essential aspects of videos, i.e. temporal patterns and static visual features, for unsupervised video event retrieval. Then to further investigate how each aspect contributes to the final hashing performance, in Chapter 6, we propose a novel information filtering mechanism called Adaptive Selection, which exploits the complementary advantages of the above two aspects for supervised video event retrieval.

Chapter 2

Literature Review

In this chapter, we review the related work on video event classification and retrieval. Firstly, in Section 2.1, related work on video event classification regarding low-level video representations, high-level concepts, weak semantic relevance and temporal-aware attention models is introduced. Secondly, in Section 2.2, we review the related work on video event retrieval with respect to video hashing, learning to hash methods, utilizing temporal patterns and static feature and temporal pattern modelling.

2.1 Video Event Classification

Video event classification is widely applied in many real-world applications, such as security surveillance and human-computer interaction, and is fast becoming one of the most significant research problems in the computer vision, multimedia, and artificial intelligence communities [9, 30, 41, 43, 58, 101, 117, 120, 126, 128]. Many classification methods have been proposed, such as methods focusing on feature representations [62, 77, 85, 96], and those focusing on classification model learning [51,52,61,63,127]. The large intra-class variation in visual content is a major challenge for video event classification. On the one hand, most of the events (e.g. "birthday parties" or "wedding ceremonies") have unconstrained content which includes various entities (e.g. objects, people, animals) with diverse interactions. And on the other hand, even videos recording the same type of event, may be very different in visual appearance. For instance, with regard to "repairing an appliance", the appliance could be a television in black or a washing machine with white paint. These kind of differences lead to large intra-class variation in visual content even within each event category. To alleviate this intra-class variation issue, methods utilizing high-level concepts [34, 94] have been proposed. There are also some methods that employ semantic information [23, 89]. Note that owing to the temporal nature of videos, the temporal information needs to be carefully modelled [72, 93]. In this section, a representative range of related work is briefly reviewed.

2.1.1 Low-level Video Representations

One direction of previous research focused on designing low-level feature representations, such as visual appearance features [13, 14, 69] and motion features [35, 42, 53, 105]. First, local features of frames or segments in a video are extracted. Then the local features are pooled into a global vector to represent a video by an encoding or pooling procedure [1,37,81,90]. The global vector representation is compact and efficient, but some important local information is neglected. It fails to exploit the underlying rich semantic information in the events, thus leading to unsatisfactory performance, and it cannot provide a semantic explanation for the classification result. Some researchers also tried to model the relationships between local features based on temporal structure [78, 97, 108].

CNN features have been used to achieve outstanding performance in many tasks, such as image classification [47, 88], image retrieval [2] and video classification [45]. In [20], it is proven that CNN features can capture the underlying semantic information of an image more accurately than traditional low-level features. Inspired by this result, in this thesis, we use CNN features as low-level representations of each frame in a video. Differing from [115], which encodes CNN features of all frames in a video into a global vector representation, in Chapter 3 we try to discover the latent concept of each frame and local segment in an event video. Ramanathan *et al.* [83] proposed the idea of learning an embedding on top of the CNN features. They focused on learning a frame representation which can capture the semantic and temporal context in a video, while the aim of our work in Chapter 3 is to adaptively abstract semantic information into hierarchical latent concepts to facilitate video event classification.

2.1.2 High-level Concept Utilization

Recently, some researchers attempted to alleviate the high intra-class variation issue by utilizing high-level semantic concepts [34, 83, 94, 119, 122]. Utilizing high-level concepts for complex video event recognition is intuitive. From low-level features to high-level concepts, it also follows the biological hierarchical structure of the visual cortex [48]. One research line of exploiting concepts is to develop a two-stage framework. First, several concepts are pre-defined and concept detectors are trained on labelled data. Then, a classification model is built on the response scores of these concept detectors. In [34], 62 concepts were defined manually. Then, 62 concept detectors were trained on manually labelled data. Based on the response scores of these concept detectors, the occurrence of each concept and co-occurrence of each pair of concepts in a video was modelled by a latent SVM framework [21]. In [94], Sun *et al.* also developed a framework based on these pre-trained concept detectors. In their method, the temporal transitions between concepts in a video were modelled by a Hidden Markov Model (HMM). Based on this generative HMM, a video was encoded into a fixed length vector by a Fisher Vector [81]. Also, Bhattacharya *et al.* [4] used a linear dynamic system to capture the temporal dynamics between the pre-defined concepts. In contrast to these two-stage methods which rely on pre-defined concepts, in Chapter 3, we propose discovery of the latent concepts for complex event classification. Using this approach, no concept database needs be constructed and maintained, thus our model does not encounter the problem of errors propagating from pre-trained concept detectors to the event classification model. Moreover, in these other methods, the capacity to handle the high intra-class variation is highly dependent on the generalization ability of the pre-trained concept detectors, whereas, our model addresses the high intra-class variation problem by adaptively abstracting semantic information into latent concepts. In [52], the authors proposed to infer binary labels of instances (i.e. frames or segments) in the videos for event classification. The binary label indicates if its corresponding instance is related to the target event. Our model extends binary labels to latent concepts which are more event discriminative. Moreover, the relationship between instances is exploited in our model, whereas, the model in [52] assumes that the instances in a video are mutually independent.

2.1.3 Weak Semantic Relevance

The major challenge to complex video event classification is the high intra-class variation caused by unconstrained content and the variety in visual appearance. To alleviate this issue, methods utilizing semantic information have been proposed [4, 34, 39, 40, 83, 94, 99]. However methods based on human-labelled semantic information [4, 34, 94] require a large amount of human effort to create and maintain a semantic information database. Alternatively, in some recent work [23, 89], methods exploiting Web-knowledge were proposed for zero-shot video event classification. These methods harvest semantic relevance from Web-knowledge, which is then utilized by applying heuristic algorithms. Jain *et al.* [36] used ImageNet objects to encode unseen video classes via semantic embedding. Gan *et al.* [22] fine-tuned a CNN that was pre-trained on ImageNet for video event classification and evidence recounting. In [23, 71], Web images related to events were collected from Google and Flickr by directly searching the event names. The authors assume these Web images have a high relevance to their corresponding events, and thus can be used to fine-tune CNNs for video event classification. In [112], CNNs pre-trained on object and scene classification tasks were, respectively, applied to videos. The probabilistic outputs of these CNNs were considered as semantic relevance with respect to objects and scenes, respectively, which were further used as the input features to a fusion network. Chang *et al.* [10] sorted the video shots by their semantic relevance, based on which an isotonic regularizer was developed to exploit the ordering information. In contrast to the above related work, in Chapter 4, we use semantic relevance generated from Web-knowledge as a weak guidance to our proposed attention model, where an attention score will be assigned to the current video shot in each timestep. The whole process is automatic without requiring any human interference.

2.1.4 Temporal-aware Attention Models

Video events contain lots of temporal information. For example, the event “birthday party” typically consists of the following activities in sequence: “people singing”, “blowing out candles”, “applauding”, and “cutting cake”. Unfortunately, this valuable temporal information is usually neglected by traditional methods (e.g. BoW) for video event classification. In Chapter 4, we use LSTM [31] to capture the temporal information in complex events. LSTM is a type of recurrent neural network (RNN) [31], which memorizes useful patterns of previous observations to provide long range context

Table 2.1: Pros and cons of video event classification methods.

Methods	pros	cons
Our method in Chapter 3.	Semantic and temporal information is exploited. Data-driven concepts.	Model training is needed. Fined-grained annotations are not used.
Our method in Chapter 4.	Semantic and temporal information is exploited. No human labor is need for collecting weak semantic relevance. Fined-grained annotations are utilized.	Model training is needed.
Conventional method with low-level features [13, 14, 35, 42, 53, 69, 105].	Compact representation. Straightforward.	Local information loss. Temporal information cannot be comprehensively utilized. Semantic gap exists.
Conventional method with pre-defined concepts [34, 94, 119, 122].	Semantic information is utilized. Some local information is reserved.	Temporal information cannot be comprehensively utilized. Large human effort is need for Pre-defined concepts. Number of concepts is limited.

for the prediction of the current step. There are many applications of LSTM such as sentiment analysis, machine translation and image captioning [95, 104]. There are also some recent works [72, 93] that use LSTM to model the temporal information in videos. However, semantic information is not used in these works. In Chapter 4, we incorporate fine-grained semantic information with a novel temporal attention model.

Attention models [76] were recently introduced for image and video captioning tasks [114, 121, 124]. In their models, the current caption word is generated by paying attentions to different image regions or different video shots in each timestep. The attention models they proposed are guided only by the ground truth of the captions. In contrast to these traditional attention models, in Chapter 4 we design a novel temporal attention model, which is not only supervised by the video-level ground truth labels but also takes into account the semantic relevance as a weak guidance to generate attention scores. The proposed attention model aims to maximize the utility of the weak semantic relevance to assist in video event classification.

Finally, we briefly summarize the pros and cons of our video event classification methods and other state-of-the-art methods in Table 2.1.

2.2 Video Event Retrieval

Large-scale content-based visual retrieval has drawn wide attention from computer vision, multimedia and deep learning communities [32, 70, 107, 116–118]. It is a challenging task which aims to retrieve the most relevant items from a large-scale visual content database accurately and efficiently. Besides conventional indexing methods [56, 107], hashing-based methods are more and more popular for content-based visual retrieval due to the following advantages. Firstly, because high-dimensional data with visual content are encoded into compact binary codes, storage cost can be reduced significantly. Secondly, based on binary codes, the retrieval task is performed efficiently in the Hamming space thanks to the high efficiency of the *xor* operation.

Generally, hashing methods can be categorized into two main kinds: data-independent methods [6, 24, 82] and data-dependent methods. The latter are also known as learning to hash methods [25, 64, 68, 110], and they can utilize some given information of the training data, such as labels (supervised) or local pairwise sample similarities (unsupervised), hence they can typically achieve better performance. In this thesis we focus on learning to hash methods for video retrieval. Many learning to hash methods [25, 49, 67, 79, 87, 106, 131] have been proposed and applied to image retrieval tasks, such as principal component analysis based hashing [25, 64], spectral hashing [110] and anchor graph-based hashing [68]. Recently, with the rise of deep learning, it has been found that deep learning to hash techniques are a great fit for image retrieval tasks [8, 50, 60, 65, 66, 113]. They unify feature learning and hash function learning seamlessly.

2.2.1 Hashing for Video Retrieval

Due to the diversity and complexity of video content and the challenging temporal nature of videos, limited video hashing methods have as yet been proposed. Some approaches discard the temporal information in video, and only utilize the static visual appearance at frame-level. For instance, the hashing methods in [7, 92] focus on frame feature pooling or learning hash codes at the frame-level, yet they neglect the temporal patterns in videos. Yu *et al.* [126] designed metrics to select some frames with discriminative visual appearance to represent a video. Furthermore, Ye *et al.* [123] proposed a supervised structural learning framework to exploit pairwise frame order, but this method only generates frame-level hash codes. In contrast to the aforementioned methods, our models in Chapter 5 and Chapter 6 take temporal patterns in videos into consideration and directly produce video-level hash codes.

2.2.2 Learning to Hash

Many learning to hash methods [25, 49, 67, 79, 87, 106, 131] have been proposed and perform well on image retrieval tasks. They are more popular than data-independent hashing methods [6, 24, 82]. The former learn their hash functions from data and can exploit useful information carried by the data such as semantic labels and local pairwise sample similarities, thus achieving better performance than data-independent hashing methods. We refer readers to [107] for a comprehensive survey on learning to hash methods.

By leveraging deep learning techniques, deep learning to hash methods [8, 50, 60, 65, 66, 113] boost the image retrieval performance even higher. Firstly, deep learning to hash leverages powerful convolutional neural networks (CNNs) to transform the original RGB image data into real-valued features. Then, these features are fed to hash functions, which are typically implemented with fully connected layers and a final quantization layer, to generate binary codes. In the training phase, the features and the hash functions are simultaneously learned.

The proposed models in Chapter 5 and Chapter 6 both have a deep architecture, so from this point of view, our methods are also related to recent work on deep hashing [17, 50, 60, 65]. For example, Lai *et al.* [50] proposed a deep network for simultaneous feature learning and hash coding with triplet ranking loss. Li *et al.* [60] incorporated pairwise labels for deep hashing. The above deep hashing methods focus on image hashing and cannot utilize the temporal information in videos. In contrast, our models are designed for jointly and adaptively modelling static visual appearance and temporal patterns to facilitate video hashing.

2.2.3 Utilizing Temporal Patterns for Video Hashing

Recently, Zhang *et al.* [130] proposed a binary LSTM (BLSTM) for unsupervised video hashing. BLSTM focuses on modelling the temporal patterns in videos by using LSTM with binarized cells. It approximates the independence and balance constraints on hash codes by using batch normalization [33] on the cell state in LSTM. In contrast to BLSTM, besides exploiting video temporal patterns, our method in Chapter 5 jointly models temporal patterns with static visual appearance. Furthermore, our model imposes the above constraints directly and strictly on the hash codes without relaxation.

Motivated by recent advances in deep representation learning [46, 95, 103], in Chapter 5 and Chapter 6, we employ deep learning to hash for video retrieval. To model the temporal patterns in videos, we use LSTM as our temporal encoder. LSTM is a type of recurrent neural network (RNN) [31] for temporal sequence modelling. In [93], Srivastava *et al.* developed an unsupervised video representation learning framework by using LSTM for action recognition. Whereas, we use LSTM as a temporal encoder to preserve temporal information for video hashing.

2.2.4 Static Feature and Temporal Pattern Modelling

In contrast to the flourishing image hashing methods, limited learning to hash methods have been proposed for video hashing, due to the diverse and complex nature of video content and the challenging temporal nature of videos. Some methods only exploit static visual features [7, 92, 126] by focusing on frame feature pooling or generating frame-level hash codes. Information loss is inevitable during feature pooling or then integrating frame hash codes to create video hash codes. In addition, the temporal patterns in videos are ignored in these methods. Ye *et al.* [123] developed a structural learning framework to utilize pairwise frame order but not intact temporal information. Again, this method only generates frame-level hash codes. In contrast to the aforementioned methods, our frameworks in Chapter 5 and Chapter 6 take both static visual feature and temporal patterns into consideration for producing video-level hash codes.

Table 2.2: Pros and cons of video event retrieval methods.

Methods	pros	cons
Our method in Chapter 5.	Jointly modeling semantic and temporal information.	Concatenated encoders. No interaction between two encoders.
Our method in Chapter 6.	Jointly modeling semantic and temporal information. Adaptively Integrating. Interaction between two streams.	More training time for LSTM layers.
Hashing only with static visual features [7,92,126].	Semantic information is utilized.	Temporal information cannot be comprehensively utilized. Local information loss.
Hashing only with temporal patterns [123,130].	Temporal information is utilized.	Semantic information is not well explored. More training time for LSTM layers.

To utilize temporal information, Zhang *et al.* [130] employed binary LSTM (BLSTM) for self-supervised video hashing. They concentrated on extracting higher-level temporal patterns from frames by using LSTM with binarized cells, while lower-level static visual features were neglected. In Chapter 5, we propose to jointly model the static visual features and temporal patterns in videos. For joint modelling, we impose learning constraints on the outputs of the temporal encoder and the static visual encoder. Joint learning of the two aspects is implemented at loss-level. To further investigate how each aspect contributes to the final hashing performance, in Chapter 6 we design a dual-stream network architecture where adaptive modelling of the two aspects happens at the hash layer equipped with a novel Adaptive Selection mechanism. We aim to learn an integrating function to maximize the utility of the two aspects automatically.

LSTM [31] has shown its strength in many sequence modelling tasks, e.g. sentiment analysis, machine translation and image captioning etc. [95,104]. In contrast to other neural networks, LSTM has a gating mechanism, which controls the information flow path in the network and enables long-term memory. Parallel to our work in Chapter 6, Miech *et al.* [73] utilize a gating mechanism to recalibrate the activations of the input representations for video classification. Inspired by the gating mechanism and the gated linear unit [16] (GLU, introduced recently for language modelling), in Chapter 6 we propose an Adaptive Selection mechanism to control which components in each aspect, i.e. static visual feature and temporal pattern, should be selected for generating video hash codes.

Finally, a briefly summary of the pros and cons of our video event retrieval methods and other state-of-the-art methods is in Table 2.2.

Chapter 3

Hierarchical Latent Concept Discovery for Video Event Classification

One of the most important tasks of video event understanding is classification. Conventional classification methods typically focus on low-level features. However, a semantic gap exists between the low-level features and the high-level video events. And the temporal information in videos can not be fully exploited by low-level features. In this chapter, we will explore how to jointly utilize semantic and temporal information for video event classification.

3.1 Introduction

One common approach to complex video event classification is to use low-level features directly. A typical practice of this approach is the Bag-of-Word (BoW) representation. It is straight forward yet achieves a reasonable result [41]. Typically, BoW follows three steps: First, low-level features (e.g. SIFT [69], STIP [53], Dense Trajectory [105]) are extracted from frames or short segments of a video. Then, the low-level features from the video are pooled into a fixed-length feature vector. Finally, fixed-length feature vectors are fed into classifiers such as SVM. There are some limitations to this approach. First, complex events possess rich semantic cues which we refer as to "concepts". For example, "a wedding ceremony" can be detected by several concepts such as "church", "presenting ring", and "bride holding flowers". But integrating low-level features of frames or short segments

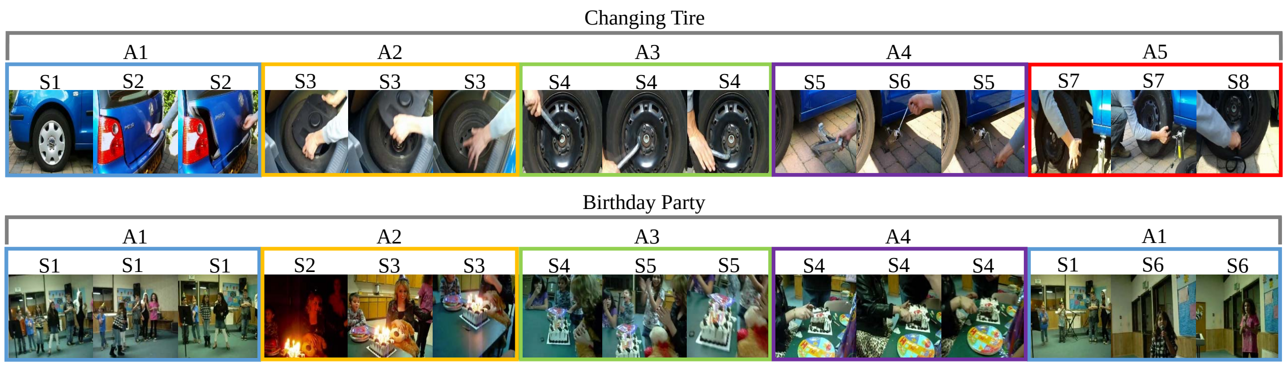

FIGURE 3.1: Demonstration of static-visual concepts and activity concepts in event videos. The example on top is a video of event "Changing Tire", and bottom example is a video of event "Birthday Party". S1–S8 indicate static-visual concepts. A1–A5 represent activity concepts. All of these concepts are abstractions of semantic cues. Along the time line, "Changing Tire" has 5 different activity concepts and "Birthday party" has 4 different activity concepts (A1 first appears at the the beginning of the event video and then presents again at the end).

into a feature vector may fail to preserve these concepts and without these concepts, a semantic explanation of the classification result cannot be provided. Second, low-level features are pooled in a pre-defined manner, thus temporal information is basically dropped. Finally, pooling features from video frames or segments which are uninformative to the specific event(s) introduces noise into the final representation. Recently, use of Convolutional Neural Networks has achieved ground breaking success in Computer Vision literature [45, 47]. CNN features have been proven to be able to capture the semantic information of an image more accurately than traditional low-level features can [20]. Therefore, however, there can be thousands of frames in a video of even two minute length. How to utilize CNN features derived from these frames for video event classification is an open question.

Another approach is to utilize the rich semantic cues (i.e. concepts) [120,132,133] in the videos via two-stage frameworks [4, 34, 94]. In the first stage, several concepts are defined, which are treated as basic components of complex events. Concept detectors are trained on low-level features of data from other domains, in which concept labels are available (we refer to these kind of concepts as extrinsic concepts), and then these concept detectors are applied to video segments (fixed-length segment or sliding windows) [34,94]. In the second stage, a classification model is built on the response scores of concept detectors, such as a Fisher Vector [94], a latent SVM [34], and a linear dynamic system [4]. It was no surprise to discover that, by employing semantic information they all outperformed the BoW approach. One limitation of this kind of approach is that it requires a large effort to train pre-defined concept detectors and the input of expert knowledge with regard to different domains. And when a new event comes along, it may require new concepts to be defined. Furthermore, the errors from the first stage may propagate to the second stage, and therefore the final event classification performance depends critically on the quality of the pre-defined concepts and the pre-trained concept detectors in the first stage. If the pre-defined concepts do not appropriately reflect the essence of the events, they become distractors instead of contributors. Moreover, the errors from concept detectors severely mislead the classification model in the second stage.

Inspired by the above facts, for event classification, we propose to exploit latent concepts in the complex events with a unified hierarchical model. The latent concepts are intrinsic to the events in contrast to the pre-defined concepts whose corresponding detectors are trained from other domains. In this work, we are trying to automatically discover the latent concepts which capture the semantic cues of events, instead of maintaining a pre-defined concept database. Hence our model does not suffer from the error propagation problem aforementioned. In our model, we define two levels of concepts in complex event videos. As illustrated in Figure 3.1, one is static-visual concept and the other is activity concept. The static-visual concept lies at the frame-level, which corresponds to occurrence of some entities. For example, birthday cake presents in some frames of a birthday party video. The activity concept lies at the segment-level, which corresponds to the primitive interaction of some entities within a short video segment, such as "blowing candles out", and "cutting the cake". These two levels of concepts are represented as latent nodes and organized in a hierarchical structure in our model and the states of these latent nodes correspond to different latent concepts. The high intra-class variation of events comes from various visual semantic cues, however this variation can be alleviated by adaptively abstracting semantic cues into static-visual concepts and activity concepts which are more compact representations of videos, leading to better event classification performance.

The major contributions of this work are summarized as follows:

- We propose to discover latent concepts for complex event classification via a novel latent hierarchical model. This model alleviates the high intra-class variation in each event by adaptively abstracting semantic cues into static-visual concepts and activity concepts. In contrast to two-stage frameworks, it is a unified model and it requires no effort to maintain a concept database. The latent concepts discovered by this model are intrinsic to the event(s), hence it does not suffer from the problem of error propagation from stage one to stage two as occurs in the two-stage frameworks.

- In an intuitive and natural way, two levels of latent concepts are proposed in our model. One is a frame-level static-visual concept, while the other is a segment-level activity concept which captures the temporal relationships between static-visual concepts within the corresponding segment. The proposed hierarchical model is more event discriminative as compared to frameworks which use only one level of concepts. Furthermore, we develop an efficient alternative linear programming algorithm for latent concept inference.

- We conduct extensive experiments on four challenging datasets, i.e. MED11, CCV, UQE50 and FCVID, and compare the performance with the state-of-the-art approaches. Our method outperforms others by large margins. Our experimental study also demonstrates the effectiveness of the proposed model with respect to latent concepts discovery.

3.2 The Proposed Model

In this section, we develop a hierarchical model to discover latent concepts for video event classification. Latent variables that respectively correspond to static-visual concepts and activity concepts are organized into a hierarchical structure. The underlying semantic information is abstracted into latent concepts adaptively by the proposed model.

3.2.1 Latent Concepts and Hierarchical Structure

In our model, we define two levels of latent concepts, i.e. static-visual concepts and activity concepts. In a real event, multiple entities may exist. The occurrence of entities and the interactions between them constitute a specific event. Take a child's birthday party as an example. The entities include kids, parents, birthday cake, etc. The occurrence of kids and birthday cake at one particular moment is a static-visual concept, and the primitive interactions between them within a short period of time is an activity concept, such as “parents cutting birthday cake”, “child blowing candles out”, and “parents and kids applauding”. In fact, within a short period of time, the transition and evolution of static-visual concepts constitute an activity concept, and then a series of activity concepts compose the final event. To discover these latent concepts, it is very natural to organize static-visual concepts and activity concepts in a hierarchical structure.

In the proposed model, we use CNN features [47] to represent each frame of the event videos, because of their capacity to preserve the underlying semantic cues [20] of images. Latent variables are employed to represent latent concepts. As shown in Figure 3.2, the first hidden layer corresponds to static-visual concepts, and the second hidden layer represents activity concepts.

We assume events are composed by static-visual concepts and activity concepts, and these concepts are organized in a hierarchical structure. The benefits of the proposed hierarchical structure for video event classification are twofold. On the one hand, the hierarchical structure incorporates multiple layers of intermediate semantic representations (i.e. static-visual concepts and activity concepts), which can effectively bridge the semantic gap between low-level visual features and high-level events. On the other hand, the relationships among different layers (i.e. events, concepts, visual features) are well captured through the hierarchy. In this way, the event semantics are effectively utilized to guide the process of automatically discovering the event-specific concepts, which correspond to the common semantics shared by the videos of the same type of event. Meanwhile, the influence of those video contents irrelevant to the event will be suppressed due to the lack of strong correlated semantics.

3.2.2 Model Formulation

The training set consists of N labelled videos (x^l, y^l), $l \in [1, N]$, where label $y^l \in \{-1, 1\}$ and each video $x^l = (x^l_1, x^l_2, \dots x^l_{M^l})$ consists of M^l frames. x^l_i is the feature extracted from the i-th frame. x^l is divided into K^l segments, and each segment s_k includes T frames, $k \in [1, K^l]$. In one video, we try to learn latent static-visual concepts from frames and to learn latent activity concepts from segments.

As illustrated in Figure 3.2, the x layer corresponds to frame features extracted from a video, the h^f layer corresponds to the static-visual concept of each frame, and the h^s layer represents the activity concept of each segment. x are observations, and h^f and h^s are latent variables. The state of a latent

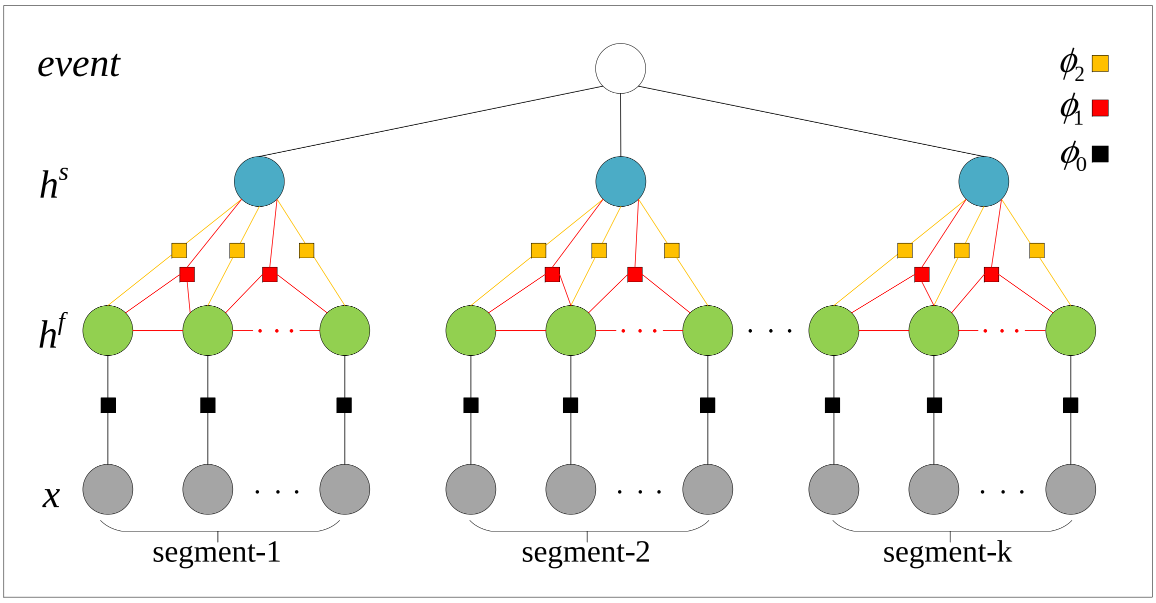

FIGURE 3.2: Illustration of the proposed model. Each circle corresponds to a variable (gray circles are observed frame features, green circles represent latent static-visual concepts, and blue circles correspond to latent activity concepts) and each square corresponds to a potential in the model. Potential $w_0 \cdot \phi_0(x_i, h_i^f)$ measures the compatibility between a frame and its corresponding static-visual concept. Potential $w_1 \cdot \phi_1(h_i^f, h_{i+1}^f, h_k^s)$ models the correlation between static-visual concepts. It measures the compatibility between two static-visual concepts of two neighbouring frames and the activity concept of their parent video segment. Potential $w_2 \cdot \phi_2(h_i^f, h_k^s)$ measures the compatibility between the static-visual concept of a frame and the activity concept of the video segment to which the frame belongs. It captures the occurrence information of static-visual concepts.

variable indicates the type of latent concept. We use H^f and H^s to denote the set of all possible states of static-visual concepts and activity concepts respectively.

Given the frame features x of an event video, the discriminative function used to score this video is defined as follows:

$$f_w(x) = \max_h \; w \cdot \Phi(x, h) \tag{3.1}$$

The potential function $w \cdot \Phi(x, h)$ is defined as:

$$\begin{aligned} w \cdot \Phi(x, h) =& \frac{1}{N_f} \sum_i w_0 \cdot \phi_0(x_i, h_i^f) \\ &+ \frac{1}{N_p} \sum_k \sum_{i \in s_k} w_1 \cdot \phi_1(h_i^f, h_{i+1}^f, h_k^s) \\ &+ \frac{1}{N_f} \sum_k \sum_{i \in s_k} w_2 \cdot \phi_2(h_i^f, h_k^s) \end{aligned} \tag{3.2}$$

in which $w = [vec(w_0), vec(w_1), vec(w_2)]$ is the concatenated parameter vector, and $vec(\cdot)$ is the operator to transform a matrix to a vector. $w_0 \in \mathbb{R}^{|H^f| \times d}$, $w_1 \in \mathbb{R}^{|H^f| \times |H^f| \times |H^s|}$ and $w_2 \in \mathbb{R}^{|H^f| \times |H^s|}$ represent the parameter matrices corresponding to ϕ_0, ϕ_1 and ϕ_2 respectively. ϕ_0, ϕ_1 and ϕ_2 are one-hot features, which represent the configurations of the latent variables (h^f, h^s). They have the same dimensionalities as w_0, w_1, and w_2, respectively. N_f and N_p are the scale factors, where N_f denotes the frame number, and N_p is the number of the potential factor ϕ_1.

Unary Static-visual Concept Potential. Potential $w_0 \cdot \phi_0(x_i, h_i^f)$ measures the compatibility between a frame and its corresponding static-visual concept, i.e. how likely it is that frame x_i is grouped into concept h_i^f. It is parameterized as:

$$w_0 \cdot \phi_0(x_i, h_i^f) = w_0[h_i^f] \cdot x_i \tag{3.3}$$

For compactness, we use square brackets for the indexing operation. $w_0[h_i^f] \in \mathbb{R}^d$ is the h_i^f-th row of w_0 which corresponds to the hidden state $h_i^f \in H^f$, and the inner product $w_0[h_i^f] \cdot x_i$ can be interpreted as the compatibility between video feature x_i and hidden state h_i^f.

Pairwise Activity Concept Potential. In our model, an activity concept is derived from its corresponding static-visual concepts. Intuitively, the occurrence of static-visual concepts and transitions between them constitutes an activity concept. Potential $w_2 \cdot \phi_2(h_i^f, h_k^s)$ measures the compatibility between the static-visual concept of a frame and the activity concept of the video segment to which the frame belongs. This potential captures the occurrence information of static-visual concepts. To model the correlation between static-visual concepts, we introduce potential $w_1 \cdot \phi_1(h_i^f, h_{i+1}^f, h_k^s)$. It measures the compatibility between two static-visual concepts of two neighbouring frames and the activity concept of their parent video segment. The temporal information within a local video segment is captured by this potential. These two potentials are respectively parameterized as:

$$w_1 \cdot \phi_1(h_i^f, h_{i+1}^f, h_k^s) = w_1[h_i^f, h_{i+1}^f, h_k^s] \tag{3.4}$$

TABLE 3.1: Experimental results on MED11 and CCV datasets, with regard to different settings of T, where $|HF|$=$|HF|$=10

	T=2	T=5	T=8	T=11
MED11	0.731	0.768	0.751	0.750
CCV	0.635	0.647	0.641	0.638

$$w_2 \cdot \phi_2(h_i^f, h_k^s) = w_2[h_i^f, h_k^s] \tag{3.5}$$

$w_1[h_i^f, h_{i+1}^f, h_k^s]$ is the element at the h_i^f-th row, h_{i+1}-th column and h_k^s-th page in the 3-D matrix w_1. It stands for the parameter that corresponds to hidden state $h_i^f \in H^f, h_{i+1}^f \in H^f$, and $h_k^s \in H^s$, and it measures the compatibility between hidden state h_i^f, h_{i+1}^f, and h_k^s. $w_2[h_i^f, h_k^s]$ is the element at the h_i^f-th row, and h_k^s-th column in matrix w_2. It refers to the parameter that corresponds to hidden state $h_i^f \in H^f$ and $h_k^s \in H^s$, and it measures the compatibility between hidden state h_i^f, and h_k^s.

For simplicity, we assume that activity concepts are mutually independent, and no relationships between them are captured by the model. Segment length T is empirically set to 5. Experimental results with regard to different settings of T are shown in Table 3.1. An ideal model should capture the relationships between activity concepts and segment the video adaptively and modelling the relationships between activity concepts and adaptive video segmentation are two related tasks. We leave these tasks to future research for a dedicated exploration.

3.3 Model Learning and Latent Concept Inference

In this section, we describe how to learn the model parameters from labelled training samples and how the latent concepts of a given video can be discovered. First, in Section 3.3.1, we formulate the learning problem into a max-margin framework and describe how to solve it with the Non-convex Regularized Bundle Method (NRBM) algorithm. In Section 3.3.2, we then develop an alternative linear programming algorithm to infer the hierarchical latent concepts of a given video.

3.3.1 Model Learning

The learning task is to estimate the model parameter w on the training videos. Recall that, an event video example is scored by Equation (3.1). In the optimization process, all the potentials in Equation (3.2) are maximized to generate an optimal score for a video.

Specifically, we use max-margin framework to learn the model parameter from labelled training examples by solving the following optimization problem:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}||w||^2 + C\sum_{l=1}^{N} \xi^l \\ s.t. \quad & y^l \cdot f_w(x^l) \geq 1 - \xi^l \\ & \xi^l \geq 0, \quad \forall l \end{aligned} \tag{3.6}$$

The equivalent unconstrained problem is:

$$\min_{w} \frac{1}{2}||w||^2 + C\sum_{l=1}^{N} R^l(w) \tag{3.7}$$

where $R(w)$ is the risk function:

$$R^l(w) = \max(0, 1 - y^l \cdot f_w(x^l)) \tag{3.8}$$

$$= \max(0, 1 - y^l \cdot \max_{h} \ w \cdot \Phi(x^l, h)) \tag{3.9}$$

Although the objective function is non-convex, some methods have been proposed to solve this kind of max-margin optimization problem, such as the cutting plane algorithm [102], the stochastic gradient descent [21] and the proximal bundle method [125]. In this work, we adopt the Non-convex Regularized Bundle Method (NRBM) [19] which is dedicated to the non-convex case. This bundle method relies on cutting plane technique. The cutting plane of the risk function $R(w)$ in Equation (3.9) is defined using its subgradient:

$$\partial R^l(w) = \begin{cases} 0, & if\ y^l \cdot f_w(x^l) \geq 1 \\ -y^l \cdot \Phi(x^l, h^{l*}), & otherwise \end{cases} \tag{3.10}$$

where h^{l*} are the optimum latent variables based on the model parameters w:

$$h^{l*} = \arg\max_{h} \ w \cdot \Phi(x^l, h) \tag{3.11}$$

The bundle method aims to iteratively build an increasingly accurate piecewise quadratic lower bound of the objective function [18]. Such a cutting plane $c_{w_t}(w)$ built in step t is a linear lower bound of the risk function $R(w)$ at point w_t, and $\frac{1}{2}||w||^2 + C \cdot c_{w_t}(w)$ is a quadratic lower bound of the objective function in Equation (3.7) [18]. More details about NRBM can be found in [19].

3.3.2 Latent Concept Inference

The NRBM algorithm for learning the model parameter w described in Section 3.3.1 needs an inference algorithm to find the optimal h^* for a given video example x:

$$h^* = \arg\max_h \; w \cdot \Phi(x, h) \tag{3.12}$$

If the latent variables h form a tree structure, the inference problem in Equation (3.12) can be solved exactly using the Viterbi dynamic programming algorithm. For a general graph, there is another option, i.e. linear programming [98]. In this work, we develop an alternative linear programming algorithm to infer latent concepts. We introduce variables z^f_{ia} to denote indicators $\mathbb{1}(h^f_i = a)$ for latent variables h^f_i and their values $a \in H^f$, with respect to static-visual concepts; z^s_{kb} to denote indicators $\mathbb{1}(h^s_k = b)$ for latent variables h^s_k and their values $b \in H^s$, correspond to activity concepts. Similarly, variables z^f_{iac} are introduced to denote indicators $\mathbb{1}(h^f_i = a, h^f_{i+1} = c)$ for two neighbouring latent variables h^f_i and h^f_{i+1} and their values $a \in H^f, c \in H^f$, with respect to pairwise potential. This alternative algorithm can be treated as a coordinate ascent algorithm. It alternates between the following two steps:

Step 1. Fixing h^s, optimize h^f:

$$\begin{aligned}\max_{0\le z\le 1}&\sum_i\sum_{a\in H^f} z^f_{ia}\cdot[w_0\cdot\phi_0(x_i,h^f_i=a)\\&+\sum_k\sum_{i\in s_k}\sum_{a\in H^f}\sum_{c\in H^f} z^f_{iac}\cdot w_1\cdot\phi_1(h^f_i=a,h^f_{i+1}=c,h^s_k)\\&+\sum_k\sum_{i\in s_k}\sum_{a\in H^f} z^f_{ia}\cdot w_2\cdot\phi_2(h^f_i=a,h^s_k)\end{aligned}\tag{3.13}$$

$$s.t.\quad \sum_{a\in H^f} z^f_{ia}=1,\quad \forall i \tag{3.14}$$

$$\sum_{a\in H^f}\sum_{c\in H^f} z^f_{iac}=1,\quad \forall i \tag{3.15}$$

$$\sum_{a\in H^f} z^f_{iac}=z^f_{i+1\,c},\quad \forall i,c \tag{3.16}$$

$$\sum_{c\in H^f} z^f_{iac}=z^f_{ia},\quad \forall i,a \tag{3.17}$$

Step 2. Fixing h^f, optimize h^s:

$$\begin{aligned}\max_{0\le z\le 1}&\sum_k\sum_{i\in s_k}\sum_{b\in H^s}\Big[z^s_{kb}\cdot w_2\cdot\phi_2(h^f_i,h^s_k=b)\\&+z^s_{kb}\cdot w_1\cdot\phi_1(h^f_i,h^f_{i+1},h^s_k=b)\Big]\end{aligned}\tag{3.18}$$

$$s.t.\quad \sum_{b\in H^s} z^s_{kb}=1,\quad \forall k \tag{3.19}$$

where Equations (3.14) (3.15) (3.19) capture normalization constraints [98], and Equations (3.16) (3.17) represent marginalization constraints [98].

The linear programming problem has integral optimal solution if the latent variables form a forest. For general graph topology, however, the optimal solution can be fractional. This is not surprising since the problem in Equation (3.12) is NP-hard [98]. After we decompose the original inference problem into the two-step alternative linear programming problem, the latent variables to be inferred in each step form a forest. Hence we can get an integral optimal solution, and the sub-problem in each step is easier to solve. Because of these facts, this alternative linear programming algorithm is very efficient. The details of this algorithm are in Algorithm 1.

Algorithm 1 The Alternative Linear Programming Algorithm for Latent Concept Inference.

```
Input: Video $x$ and the learned model $w$;
Output: $h^f$ and $h^s$;
1: Initialize $h^f$ and $h^s$;
2: Calculate $fval \leftarrow f_w(x, h^f, h^s)$;
3: repeat
4:     Update $h^s$ using $h^f$ by solving Equation (3.18);
5:     Update $h^f$ using $h^s$ by solving Equation (3.13);
6:     Calculate $fval \leftarrow f_w(x, h^f, h^s)$;
7: until there is no change to $fval$
8: return $h^f, h^s$;
```

3.3.3 Computational Cost

The computational cost of the proposed method comes from the iterations of NRBM. And in each iteration, NRBM invokes latent concept inference to compute the new cutting plane. Thus, the computation complexity of the proposed method can be written as $O(I_n * N * I_i * C(\bar{m}))$. I_n is the number of iterations of NRBM, N is the number of videos, I_i is the number of iterations of latent concept inference which is performed by linear programming, and $C(\bar{m})$ is the complexity of one iteration of latent concept inference for a video with $\bar{m}$ frames, where $\bar{m}$ is the average frame number per video. NRBM is a state-of-the-art method for optimizing a regularized objective with non-convex risk. The efficiency of NRBM has been shown in [19]. Refer to [19] for a detailed theoretical analysis of the algorithm. $C(\bar{m})$ depends on the solver used in the linear programming. In our experiments we use the interior-point solver. A more detailed theoretical analysis of the complexity of linear programming can be found in [5]. In our experiments, typically we observed that the latent concept inference converges within 10 iterations and the NRBM converges within 200 iterations. On the largest dataset we have used, i.e. FCVID which consists of 91,223 videos, the average learning time for one event on a single Intel Xeon CPU @2.60GHz is around 8 hours.

3.4 Experiments

In this section, we first describe the datasets and experiment settings we adopted to evaluate the proposed model. Then we conduct experiments on the proposed model and compare its performance with baseline methods as well as the state-of-the-art methods.

3.4.1 Datasets and Settings

To evaluate our model, we conduct experiments on four datasets: 1. TRECVID MED11 EventKit dataset [101]; 2. Columbia Consumer Video (CCV) dataset; 3. UQ Event dataset with 50 pre-defined events (UQE50); and 4. Fudan-Columbia Video Dataset (FCVID).

MED11. This dataset contains 2047 diverse videos collected from the internet. These videos fall into 15 events and event names are listed in Table 3.2. In order to compare our model with [34,83,94], we followed the same protocol, where 70% videos are randomly selected from MED11 EventKit for training and 30% for testing.

CCV. This dataset contains 9,317 YouTube videos covering 20 event categories. The event names and train/test splits can be found in the original paper [44].

FCVID. To the best of our knowledge, FCVID [43] is one of the largest video datasets currently available for event classification. It consists of 91,223 Web videos annotated manually into 239 categories. The total duration of all the videos is 4,232 hours and the average video duration is 167 seconds. The categories are organized into a hierarchy and we consider the root of the hierarchy to be level 1, and the most distant leaf nodes to be in level 4. We use the nodes in level 3 as final categories (a total of 57 categories including 28 leaf nodes and 29 non-leaf nodes), thus the videos belonging to leaf nodes in level 4 will be grouped together into their corresponding parent categories. In this scenario, each category contains videos from different leaf nodes. Thus, the intra-class variation with respect to the new 57 categories is much higher than it was with regard to the original 239 categories. Our method aims to discover latent concepts from a variety of video content. The higher the intra-class variation, the more challenging the video is to our method. For fair comparison, all other compared approaches are implemented using the same settings. **UQE50.** Video dataset UQE50 (UQ Event dataset with 50 pre-defined events) is new released for event analysis tasks, which contains 3,462 event videos divided into different event categories and 18,495 distractors that are irrelevant to any

TABLE 3.2: Events in MED11 and partial events in UQE50

ID	MED11	UQE50
1	Attempting a board trick	APEC Russia 2012
2	Feeding an animal	Aussie football AFL 2012
3	Landing a fish	Australian election 2013
4	Wedding ceremony	Bangladesh Factory Disaster 2014
5	Woodworking project	Beijing Olympic Opening Ceremony
6	Birthday party	Boston Marathon bombings 2013
7	Changing a vehicle tire	Brisbane festival river fire 2013
8	Flash mob gathering	Buda Wiener dog race 2012
9	Getting a vehicle unstuck	Coldplay Paradise Live France 2012
10	Grooming an animal	Costa Concordia disaster 2012
11	Making a sandwich	Curiosity rover lands on Mars 2012
12	Parade	Deepwater horizon oil spill 2010
13	Parkour	East Africa drought 2011
14	Repairing an appliance	FIFA world cup Brazil 2014
15	Sewing project	Facebook debut on Nasdaq 2012

pre-defined events. Partial event names are listed in Table 3.2. We use 2,122 event videos and 1,850 distractors as training data and 1,340 events videos and 16,645 distractors as test data. All videos in this dataset are downloaded from YouTube. The complexity of the events in this dataset is higher than that of the two other video event datasets. The videos from UQE50 are all of hot global events that occurred in the last few years and they contain far more complex patterns than the other datasets that mainly contain activity or action sequences. For instance, some events have a long duration and show complicated visual scenes like "APEC Russia 2012". Some videos record the same specific event but show different visual appearances because they are shot from different camera angles. The events defined in the UQE50 dataset are very ad hoc, and some of the events have very few positive instances. Consequently, the event recognition task on this dataset is very challenging.

We sample one frame every second for each video. Caffe [38] is used to extract CNN features from each frame. On the MED11, CCV and UQE50 datasets, we use a pre-trained model described in [47] and adopt the same pre-processing as in [47]. On the FCVID dataset we adopt a better (with respect to the image classification task on ImageNet) deep CNN model VGG-16 [88] for feature extraction. CNN features from layer fc7 [47] are used in our experiments. We employ average precision to evaluate the classification performance for a single event and mean average precision for overall performance for all events.

A Baseline Method: K-means-State. The proposed model detects an event by discovering the latent concepts of frames and segments. Discovery of the latent concepts is an adaptive learning process for each event category. One naive approach is to use K-means clustering to infer the latent concepts. We introduce this approach as a baseline method referred to as K-means-State. On the training dataset, we perform the k-means algorithm over all frames of every video and get $k = |HF|$ frame-centres. For each segment we use the average of each frame feature within it as segment feature, then we can also get $k = |HS|$ segment-centres. The closest centre is assigned to each frame or segment as its latent state. Then the feature vector $\Phi(x, h)$ in Equation (3.2) can be calculated. After we get all the feature vectors of each video, a linear SVM model is trained on them. On the test dataset, the states of frames or segments are first inferred based on frame-centres and segment-centres trained on training dataset. Then the trained SVM model can be used to perform event classification. Note that, the objective function of the proposed model is non-convex, hence local optima will possibly be reached. The SVM model trained by K-means-State is also a reasonable initialization of the proposed model.

3.4.2 Experimental Study

Size of Possible Latent States. We first investigate how the model parameter will influence the classification performance. Recall that $|H^f|$ and $|H^s|$ are the sizes of possible states of static-visual concept and activity concept, respectively. We depict the performances of our model with respect to different parameter configurations in Figure 3.3. First we set $|H^f| = |H^s| = |H|$. When $|H| = 2$, the latent concept becomes a binary variable, and it can be interpreted as an indicator which indicates whether its corresponding frame or segment is related to the event. This is a special case of the proposed model. Note that, the model (which we refer to as InstanceInfer) in [52] is also a special case of this

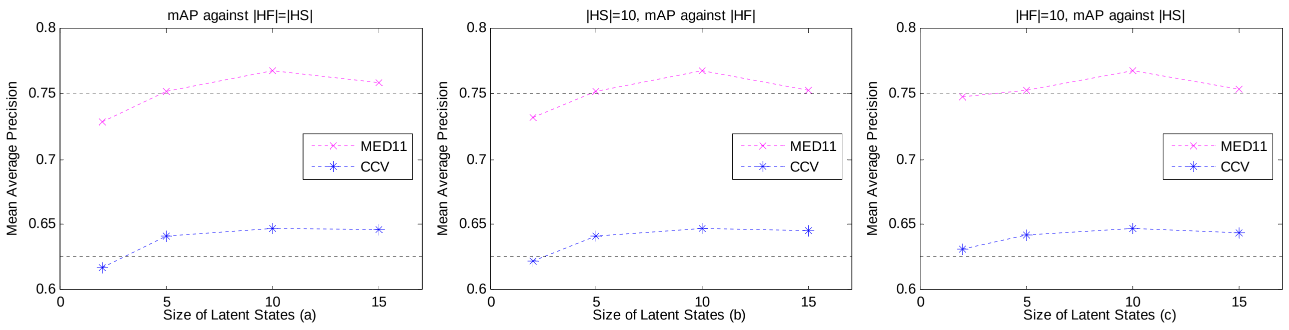

FIGURE 3.3: Performances in mean AP of the proposed model for different configurations of the size of possible hidden states. In sub-figure (a), x-axis represents $|HF| = |HS|$; in sub-figure (b), x-axis represents $|HF|$, where $|HS|$ is fixed at 10; and in sub-figure (c), x-axis represents $|HS|$, where $|HF|$ is fixed at 10.

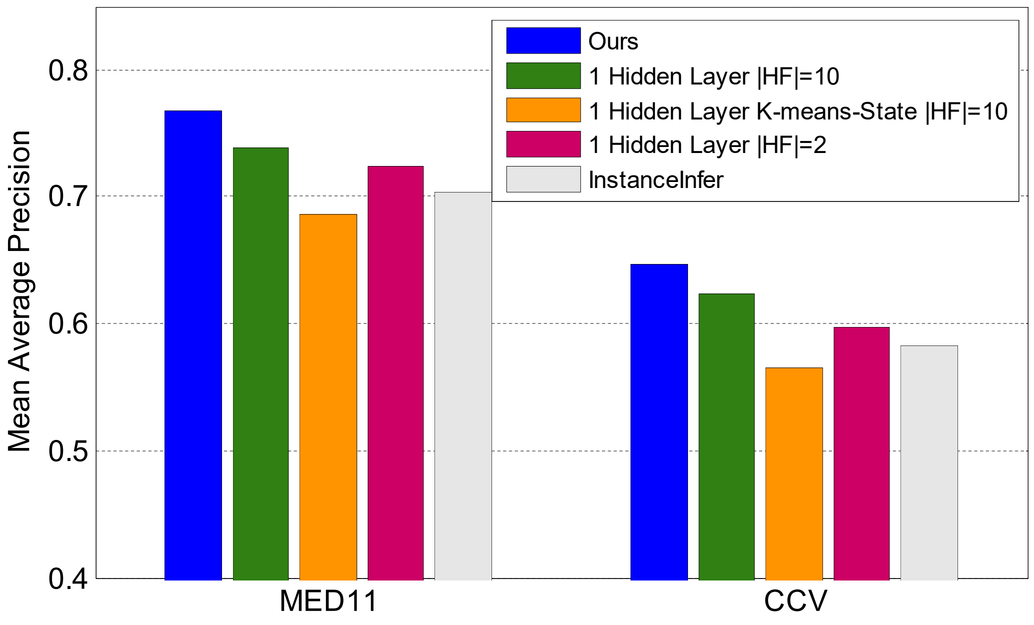

FIGURE 3.4: Mean APs of different structures of latent variables. 1 Hidden Layer $|HF|$=10 represents the variant model with only the hidden layer corresponding to frame-level static-visual concepts, where we set $|HF|$ to 10. 1 Hidden Layer $|HF|$=2 represents the same variant model where we set $|HF|$ to 2. 1 Hidden Layer K-means-State is the baseline method but performs clustering only on frames.

configuration. Although InstanceInfer infers the binary label of each instance (frame or segment) in the video, it also assumes the instances are mutually independent. However the relationships between

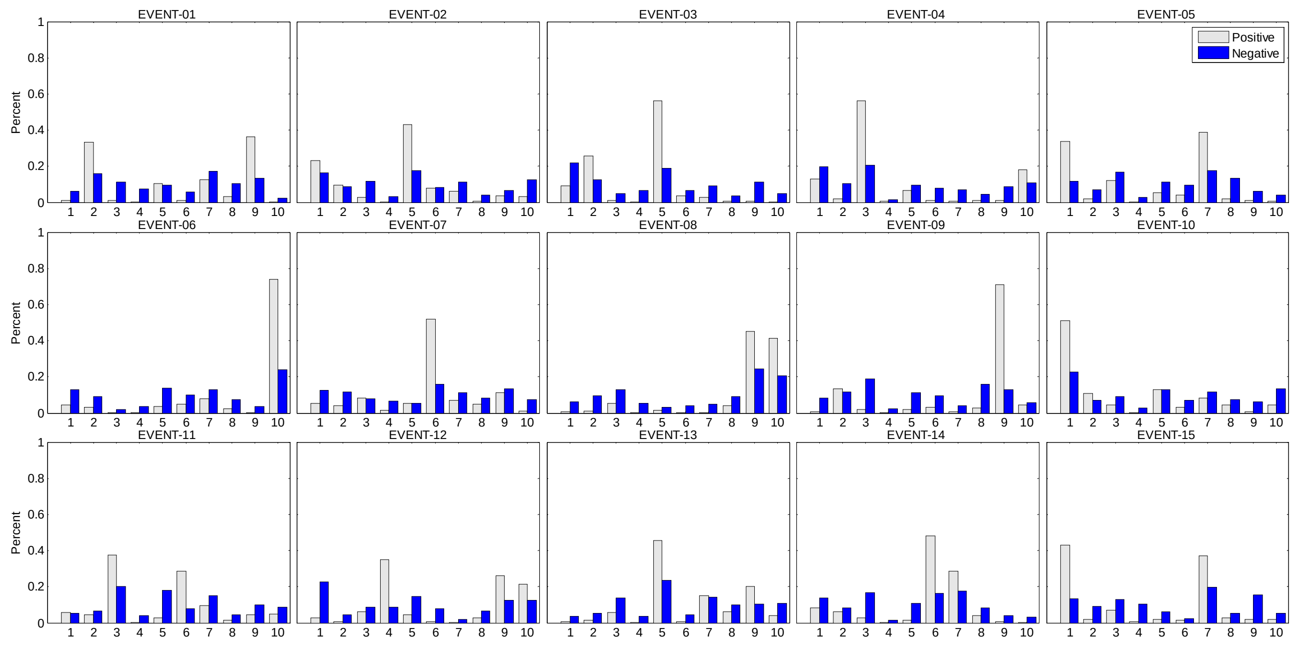

Figure 3.5: The distribution of static-visual concepts in the MED11 dataset. For each event, the distribution of the static-visual concepts from positive videos and the distribution of the static-visual concepts from negative videos is summarized separately.

instances are modelled by potential $w_1 \cdot \phi_1$ and $w_2 \cdot \phi_2$ in our model. Figure 3.3 (a) shows that the performance improves as $|H|$ increases, and a larger $|H|$ means a larger semantic capacity of the proposed model which proves that latent concepts are more event discriminative than coarse binary latent variables. We get the best performance at $|H| = 10$ both on the MED11 and CCV datasets, and when $|H|$ increases to 15, the performance drops due to overfitting. Hence $|H| = 10$ is the best choice for the MED11 and CCV datasets. Furthermore, we also depict the performance against $|HF|/|HS|$ when $|HS|/|HF|$ is fixed at 10, in Figure 3.3 (b) and (c), respectively. Unless otherwise specified, in the following experiments $|HF|$ and $|HS|$ are fixed at 10.

Hierarchical Structure of Latent Concepts.

To verify the effectiveness of the proposed hierarchical structure, we compare our proposed method to its variant with only one layer of static-visual concepts as well as a multiple instance learning method, i.e. InstanceInfer [52]. From results, reported in Figure 3.4, we can see that our method performs better than all its competitors, and the variants (i.e. $|HF| = 2$ and $|HF| = 10$) achieve superior performance to that of the InstanceInfer. This is because the proposed hierarchical

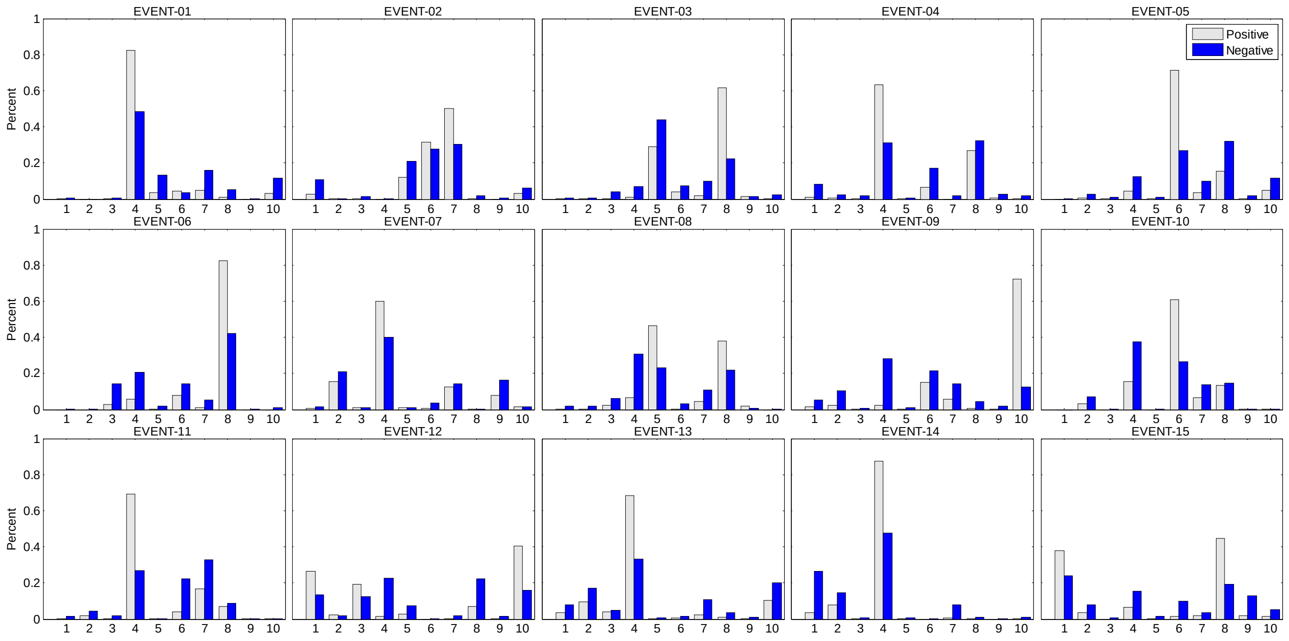

FIGURE 3.6: The distribution of activity concepts in the MED11 dataset. For each event, the distribution of the activity concepts from positive videos (grey bars) and the distribution of the activity concepts from negative videos (blue bars) are summarized separately.

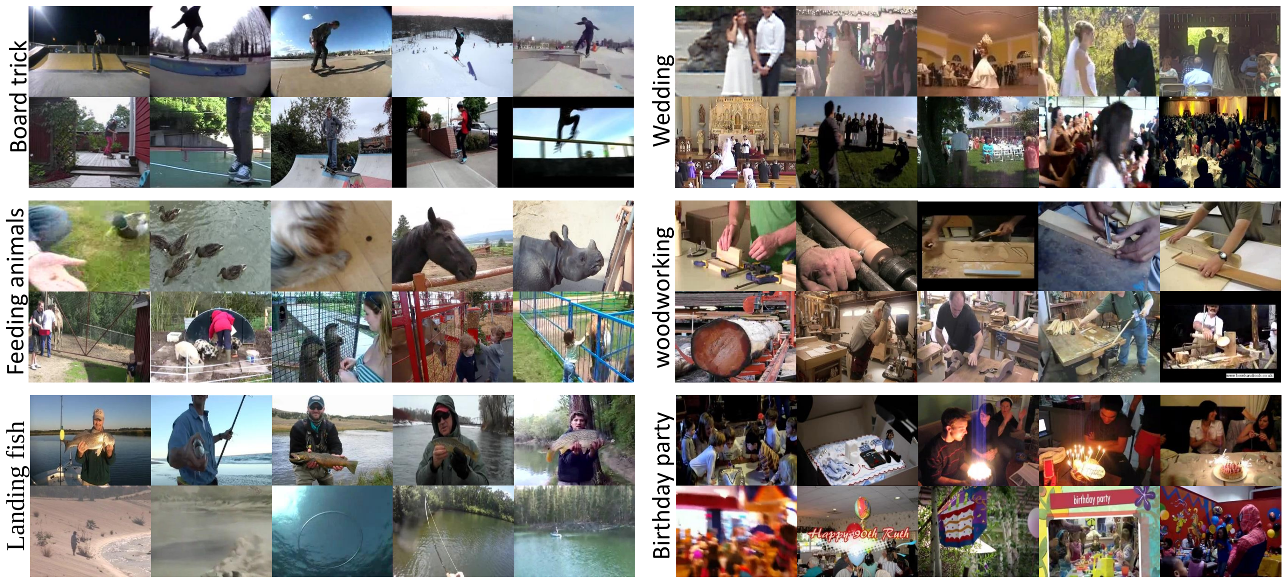

FIGURE 3.7: Instances of latent static-visual concepts on the MED11 dataset. For each event, two representative latent static-visual concepts are selected. In the top row are instances of one static-visual concept, and in the bottom row are instances of the other.

structure is able to model the nature of video data, including the relationships among frames, segments and videos, as well as explore event semantics to bridge the semantic gaps in the video event classification task.

Distribution of Latent Concepts. Once the best model configuration was determined via the above two experiments, we were curious about how the proposed model works internally. Figure 3.5, presents the distribution of static-visual concepts learned from each event from the MED11 dataset, and the distribution of activity concepts from the same dataset is shown in Figure 3.6. In these two figures, discovered latent concepts from each video are summarized into bars, and each bar corresponds to a latent concept. For each event, positive video examples (grey bars) and negative video examples (blue bars) are summarized separately.

In the two figures we can see that, for each event, the distribution of positive examples and the distribution of negative examples show very different patterns. They are very discriminative, both the static-visual concepts and the activity concepts. This means that the latent concepts are adaptively discovered by our model for each event, and semantic information from frames/segments is grouped into different latent static-visual/activity concepts which possess discriminative power for event classification. In the CCV dataset, similar discriminative distributions of latent concepts are observed, so we do not show the distributions on the CCV dataset. This result shows our model's capability to discover latent concepts and to abstract underlying semantic cues into latent concepts.

Instances of Latent Concepts. For a further exploration, Figure 3.7 shows some interesting instances of different latent static-visual concepts from different events from the MED11 dataset, discovered by the proposed model. For each event, we chose two representative latent static-visual concepts and for each concept, five frames with the highest potential were extracted from five different videos. From these concept instances we can see that frames that have similar semantic information are abstracted into the same latent concepts, even though their visual appearance varies. For example, for the event "Landing a fish", two latent concepts are shown in the figure: the first latent concept is "person holding fish with water surface as background", and the second concept is "water surface of river or lake". For the event "Wedding ceremony", the first concept is "bride and bridegroom standing together", and the second is "the people or the crowd participating in the wedding ceremony". More interesting is the event "Woodworking". The first concept is "close shot of worker's hand, tools and timber", and the second concept is "relative long shot of the work platform". All of these instances are extracted from different videos. Although their intra-class variation with regard to their visual appearance is very high, our model can successfully group them into corresponding latent concepts based on semantic cues.

Through these interesting instances, we provide an intuitive visual understanding of the latent concepts discovered by our model. The discovered latent concepts are useful in explaining the classification result. Moreover, they can be used to build a semantic-aware index for event videos, but we leave the solution to this problem for future research.

3.4.3 Comparison with the State-of-the-art Methods

In this section we compare our model with four state-of-the-art systems. For fair comparison we either quote the results published in the original papers or re-run the codes released by the authors. [34, 94] are both based on high-level concepts. About 60 concept detectors were pre-trained, and their frameworks were built on the response scores of the concept detectors. In [34], for each video, global low-level features, unary concept occurrence and the joint co-occurrence of two concepts were combined to model the event, but the temporal information among these concepts was not considered. Note that for low-level features they used multiple image and video features including SIFT, STIP, ISA [54] and MFCC. In [94], concept transitions over time were modelled by the Hidden Markov Model (HMM), then each video was encoded into a feature vector by a HMM Fisher Vector which was derived by applying a Fisher kernel [81] to the HMM. A dense Trajectory is used to train the concept classifiers. In contrast to [34, 94], our model does not depend on pre-trained concept classifiers, instead, we try to discover the latent concepts (which are intrinsic to each event) for event classification and the underlying semantic information is modelled by the proposed hierarchical structure of latent concepts. We also compare our method with another framework which is also based on CNN features. In [83], a temporal embedding is learned on the top of the CNN features from the fc6 layer, thus, capturing temporal semantic context. We use the same CNN architecture [47] as in [83] to extract CNN features. In [52], a proportion SVM model which can infer the binary label of each instance (frame or segment) in a video was proposed. The binary label indicates whether the corresponding instance is related to the target event and it assumes the instances in a video are mutually independent. Our model captures the relationships between instances and extends binary labels to

Table 3.3: Comparison of our model with other event classification methods on the MED11 dataset. The best performances are in bold font.

Event ID	Joint+LL [34]	HMMFV [94]	fc7 [83]	TE [83]	K-means-State	InstanceInfer [52]	Ours
1	0.757	**0.882**	-	-	0.731	0.776	0.856
2	0.565	0.461	-	-	0.586	**0.636**	0.608
3	0.722	0.789	-	-	0.742	0.761	**0.792**
4	0.675	0.811	-	-	0.708	0.824	**0.833**
5	0.653	0.623	-	-	0.667	0.592	**0.728**
6	0.782	**0.814**	-	-	0.752	0.748	0.799
7	0.477	0.518	-	-	0.740	0.668	**0.809**
8	**0.919**	0.877	-	-	0.828	0.788	0.824
9	0.691	0.772	-	-	0.886	0.873	**0.943**
10	0.510	**0.634**	-	-	0.484	0.610	0.620
11	0.419	0.524	-	-	0.534	**0.683**	0.680
12	0.724	**0.770**	-	-	0.711	0.671	0.700
13	0.664	**0.890**	-	-	0.553	0.618	0.781
14	0.782	0.634	-	-	0.655	0.727	**0.786**
15	0.575	0.621	-	-	0.723	0.578	**0.759**
mean AP	0.661	0.708	0.691	0.711	0.687	0.703	**0.768**

latent static-visual concepts and activity concepts.

On the MED11 dataset, we follow the same experiment protocol and quote the results published in [34,83,94]. For InstanceInfer [52], we re-run the author's code on this dataset. As illustrated in Table 3.3, our model achieves the best performance, with 6% and 10% improvement over [94] and [34], respectively, in mean AP. This comparison indicates that the latent concepts discovered by our model are better than the pre-trained concepts for event classification. In column 3, we quote the performance of linear SVM on the fc7 CNN features from [83]. Our model improves significantly upon these results, showing that the good performance is not simply because of the discriminative power of CNN features. Furthermore, our model outperforms the temporal embedding (TE) proposed in [83]

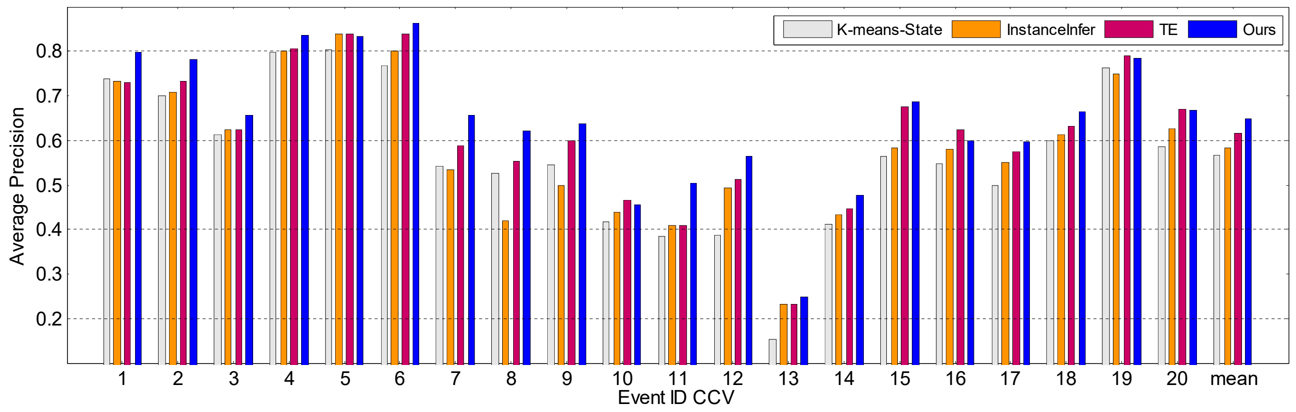

Figure 3.8: Evaluation results on the CCV dataset. The mean APs of K-mean-State, InstanceInfer, TE and our model are 0.567, 0.583, 0.617 and 0.647, respectively.

which is a state-of-the-art model learned on CNN features. Finally, we report the performances of proportion SVM, i.e. InstanceInfer [52] and the baseline method K-means-State, respectively, in column 6 and column 5. Our model significantly outperforms K-means-State indicating that the adaptive learning process for discovering latent concept is superior to naive clustering and the learned concepts contribute better than K-means centres. The proposed method also improves upon the InstanceInfer method. For event classification this indicates that: first, hierarchical latent concepts are more event-discriminative than binary labels; and second, the relationships between instances (i.e. frames and segments) captured by our model is useful.

We report experimental results on the CCV dataset in Figure 3.8 and results from the UQE50 dataset in Figure 3.9. For InstanceInfer and TE, we use the codes released by the corresponding authors. On these two datasets, the proposed model outperforms other methods as on the MED11 dataset. On the CCV dataset the mean APs of K-mean-State, InstanceInfer, TE and our model are 0.567, 0.583, 0.617 and 0.647, respectively. On the UQE50 dataset the mean APs of K-mean-State, InstanceInfer, TE and our model are 0.176, 0.213, 0.216 and 0.294, respectively. On the UQE50 dataset the proposed model achieves 38% relative improvement over InstanceInfer [52] which is a more significant improvement when compared with the relative improvements over the MED11 and CCV datasets, i.e. 9% and 11%, respectively. We analyse the reason as to why the proposed model can improve so much on the UQE50 dataset as follows. Firstly, the events in the UQE50 dataset are more complex than the events in the MED11 and CCV datasets. For example, events in UQE50 like

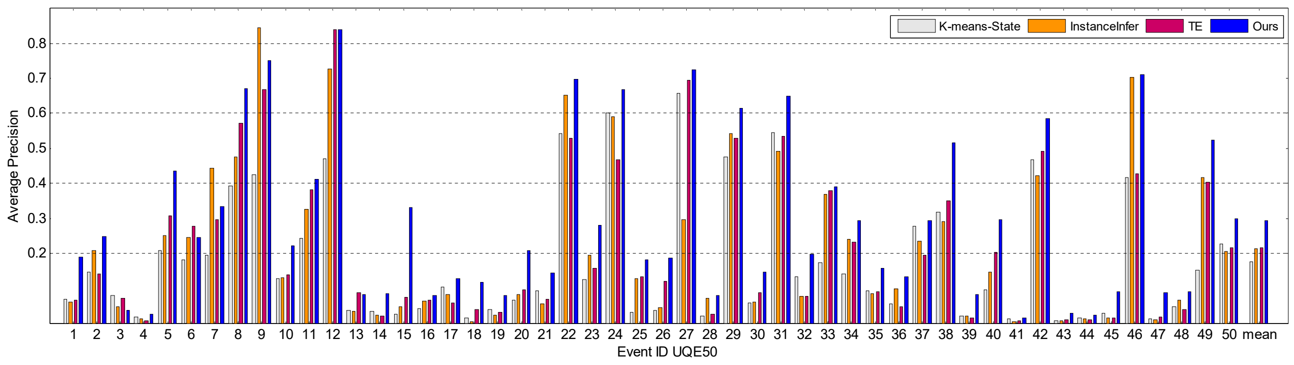

Figure 3.9: Evaluation results on the UQE50 dataset. The mean APs of K-mean-State, InstanceInfer, TE and our model are 0.176, 0.213, 0.216 and 0.294, respectively. On complex events 5 and 15, our method significantly outperforms InstanceInfer, whereas, InstanceInfer outperforms our method on events 7 and 9 which are relatively simple. This may due to the overfitting of our model on simple events and the binary labels in InstanceInfer maybe informative enough for the classification of these events.

"APEC Russia 2012" and "Beijing Olympic Opening Ceremony 2008" are more complex than events in MED11 and CCV datasets like "Wedding Ceremony" and "Feeding an animal". The intra-class variations are much higher in event videos from UQE50 than event videos from MED11 and CCV. The proposed model possesses the capacity to handle such complex event patterns because of the hierarchical latent concepts. Secondly, the proposed method exploits the relationships among latent static-visual concepts and activity concepts which are informative for complex event classification. As a case study, we investigate the performance of our method and InstanceInfer on four representative events in UQE50: "Beijing Olympic Opening Ceremony 2008" (Event ID 5), "Facebook debut on Nasdaq 2012" (Event ID 15), "Brisbane festival river fire 2013" (Event ID 7), and "Coldplay Paradise Live in France 2012" (Event ID 9). Events 5 and 15 are more complex than events 7 and 9. Complex visual patterns present in videos from events 5 and 15, but in videos from events 7 and 9 there are only relatively simple visual patterns such as "firework in sky" and "crowd of people", etc. As depicted in Figure 3.9, on events 5 and 15, our method significantly outperforms InstanceInfer as expected, whereas, InstanceInfer outperforms our method on events 7 and 9 which are relatively simple events. This may due to the overfitting of our model on relatively simple events and the binary labels in InstanceInfer must be informative enough for the classification of these events.

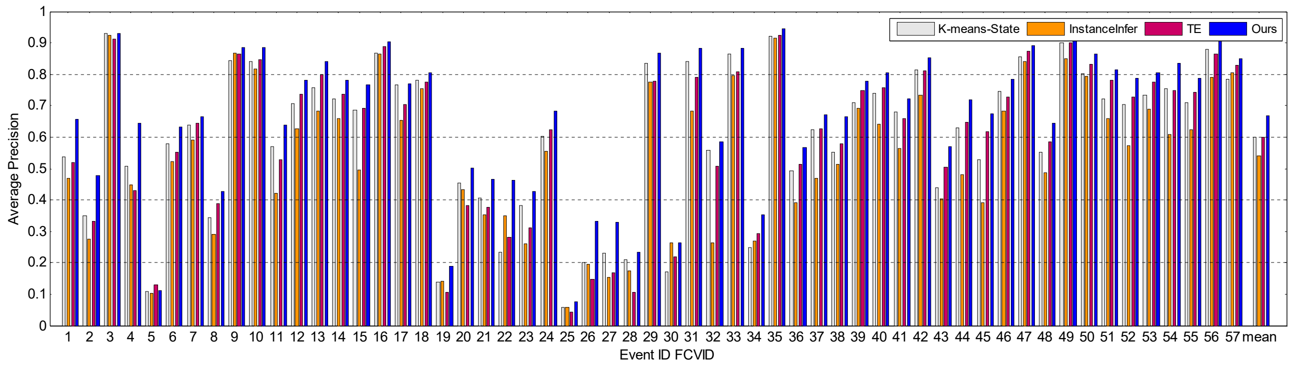

Figure 3.10: Evaluation results on the FCVID dataset. The mean APs of K-mean-State, InstanceInfer, TE and our model are 0.601, 0.540, 0.600 and 0.667 respectively.

To further verify the effectiveness of the proposed method, we also perform experiments on a large-scale video dataset, i.e. FCVID. To the best of our knowledge, FCVID is one of the largest video datasets for event classification. From the experimental results shown in Figure 3.10, we can observe that our method significantly outperforms other state-of-the-art methods.

3.5 Summary

In this chapter, we proposed the discovery of hierarchical latent concepts for video event classification utilizing underlying semantic cues. Our model abstracts the underlying semantic information into two levels of latent concepts: frame-level static-visual concepts and segment-level activity concepts. A hierarchical structure was proposed to model these latent concepts. An activity concept is comprised of a sequence of static-visual concepts, where the temporal information of static-visual concepts is captured. In this way, we utilize the temporal information and semantic information collectively and make them cooperate with each other.

A max-margin framework is employed for model learning, then we developed an alternative linear programming algorithm for latent concept inference. The experimental results show that the proposed model outperforms the state-of-the-art methods on four challenging datasets, i.e. MED11, CCV, UQE50 and FCVID. In contrast to two-stage frameworks, the proposed model requires no effort to construct and maintain a concept database, and the latent concepts are discovered adaptively based on the underlying semantic cues. Thus, our model does not encounter the error propagation problem which occurs in two-stage frameworks. Compared with InstanceInfer [52], the hierarchical latent concepts are more informative than binary labels for event classification. Furthermore, the concepts discovered by our model are not only helpful for explaining the event classification result but also can be used to build a semantic-aware index for event videos.

Chapter 4

Weak Semantic Relevance Utilization for Video Event Classification

In the previous chapter, we have formalized the semantic and temporal information into latent concepts and we have discussed how to discover the latent concepts for video event classification. In the proposed hierarchical model, the static-visual concept and activity concept are latent variables and the inference process is data-driven. The only available supervision is the event labels of videos. There are no fine-grained annotations for the latent concepts. Nonetheless, we believe the fine-grained annotations of video shots may improve the classification performance. However, it requires large human effort to obtain fine-grained annotations. To solve this dilemma, in this chapter, we propose to utilize weak semantic relevance, which can be easily acquired from the Web, to facilitate video event classification. A novel temporal attention model is designed to collectively exploit the weak semantic relevance and temporal information to filter out irrelevant video shots. Thus, it can achieves better classification performance.

4.1 Introduction

Essentially, a video consists of a sequence of shots. Generally, not all the shots are relevant to the event represented by the video and a natural way to evaluate the importance of a video shot to the event it belongs to is to exploit its semantic relevance [4, 34, 94] to the event of interest. Specifically,

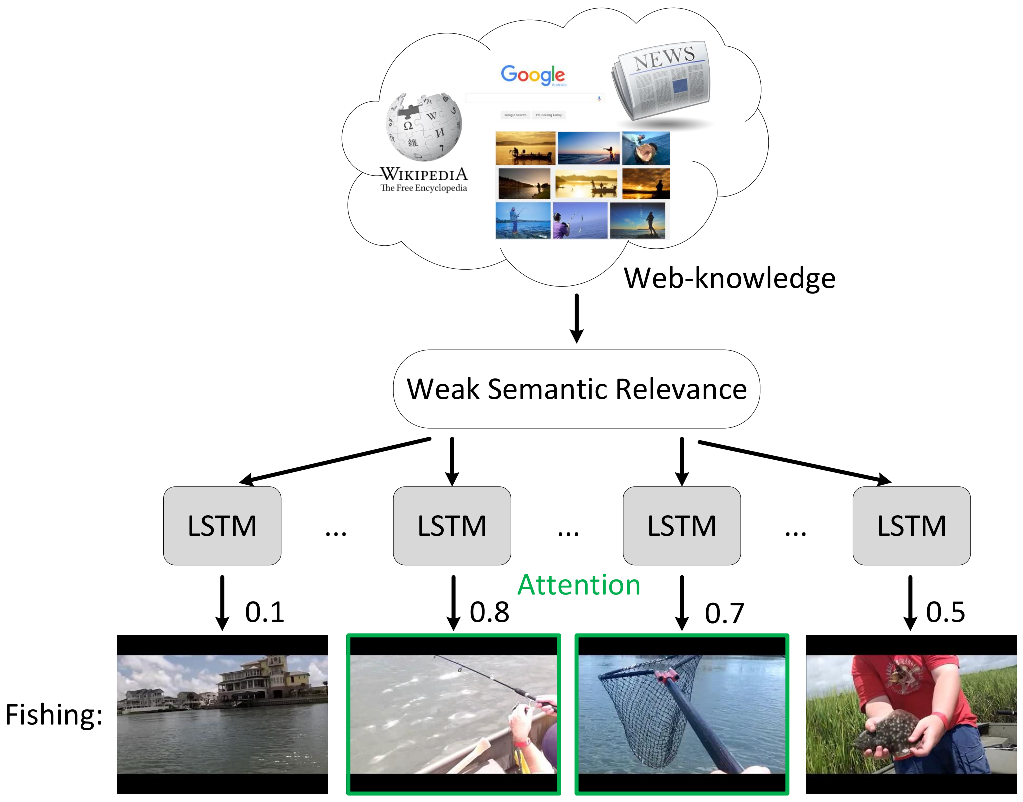

FIGURE 4.1: Illustration of the proposed framework. Our framework first harvests weak semantic knowledge from Web-knowledge, then uses it as a weak guidance to the attention model. The LSTM layer then employs the attention model to assign an attention score for each shot in a video.

when classifying a video, we wish to pay more attention to the shots with high semantic relevance to the target event, and neglect the ones with low relevance. How to assign semantic annotations to each video shot and how to measure the shot's relevance to the target event are two major research issues to address in video event classification.

In some recent works [4, 34, 94], a small number of event-related semantic concepts (less than 100) were pre-defined. Concept detectors were then trained on the manually annotated video shots and the response scores of the testing video shots with respect to these detectors are used as semantic relevance to the target event. But to prepare the annotated video training set, a large amount of human effort is required. Moreover, when a new event is introduced each video needs to be annotated again. In contrast to the prohibitive labour cost on obtaining sufficient semantic annotations for every shot in millions of videos, Li *et al.* [58] proposed to automatically discover latent concepts in a data-driven manner. Furthermore, weak semantic relevance can be conveniently gained from easily accessible Web-knowledge [23, 89]. For instance, in [23, 71], event-related Web images were downloaded from Google and Flickr by directly searching for the event names. The authors assume that these Web images have a high relevance to their corresponding events and therefore can be used to fine-tune CNNs for video event classification. In [112], CNNs pre-trained on object and scene classification tasks were respectively applied to videos. The probabilistic outputs of these CNNs are considered as semantic relevance with respect to objects and scenes respectively, which are further used as the input features to a fusion network.

Once reliable semantic relevance has been determined, a straightforward way to utilize it is to directly combine it with low-level shot features (e.g. SIFT [69], STIP [53], Dense Trajectory [105]). For example, before aggregating the shot features of a video into a global bag-of-words (BoW) vector, we can weigh them by their semantic relevance to the target event. However, the weak semantic relevance gained from Web-knowledge is not always reliable and may even be noisy due to the domain gap [89, 117] between the Web-knowledge and the videos, so directly employing it without determining its reliability does not maximize its utility. Even worse, it may introduce noise into the final representation, resulting in an inferior classification performance.

Motivated by the above facts, we propose a long short-term memory (LSTM) [31] framework (illustrated in Figure 4.1) with a novel attention model which takes semantic relevance gained from Web-knowledge as weak guidance. Attention models [76] were recently used for image and video captioning tasks [114, 121, 124]. When a caption was being generated for an image, the caption model would pay attention to different regions in each step. Inspired by their success, we design a novel attention model to automatically evaluate the weight of the current testing video shot based on its weak semantic relevance to the event of interest. As aforementioned, the semantic relevance generated from Web-knowledge is weak and noisy. Therefore, to maximize its utility, the proposed attention model assigns an attention score to the current video shot automatically in each timestep by taking the semantic relevance as a weak guidance rather than simply considering the semantic relevance as the weight of each shot. The score of a testing video to a target event will be then computed based on its weighted shots.

The main contributions of our work are summarised as follows:

- To leverage weak semantic relevance for video event classification, our framework jointly optimizes two objectives at two levels. The first one is the classification loss corresponding to the video-level groundtruth label, and the second one is the shot-level relevance loss corresponding to the weak semantic relevance.
- To maximize the utility of weak semantic relevance for video event classification, we propose a novel attention model. Instead of entirely following the weak semantic relevance, the proposed attention model takes it as a weak guidance to automatically weigh each testing video shot.
- We conduct extensive experiments on three large-scale video event datasets, i.e. MEDTest14, ActivityNet and FCVID. The experimental results demonstrate the effectiveness of the proposed framework with respect to leveraging weak semantic relevance for video event classification. State-of-the-art classification performance was achieved on each of these three datasets.

4.2 The Proposed Approach

In this section, we propose a framework for video event classification, which consists of a novel attention model to generate an attention score for each shot and an LSTM layer to capture the temporal information embedded in video shots. Importantly, the proposed attention model takes the weak semantic relevance as a guidance, where the utility of the weak semantic relevance is effectively exploited to serve the video event classification.

4.2.1 Weak Semantic Relevance Extraction

In this work we use ImageNet [84] and a publicly available NLP corpus such as Wikipedia Dump [111] as our sources of computing weak semantic relevance. The ImageNet dataset has $C = 1000$ categories, each of which comes with an entity description (e.g. laptop computer, german shepherd dog). Assume there are a number of events E in our video dataset and each event has a text description. We use a Word2Vec embedding [75] that was pre-trained on a massive natural language corpus to evaluate the semantic relevance between the ImageNet category and the target event based on their text descriptions. In Word2vec embedding, each word is embedded in a continuous vector space and two words with similar semantic meanings have a close Cosine distance in this vector space [74, 75, 80].

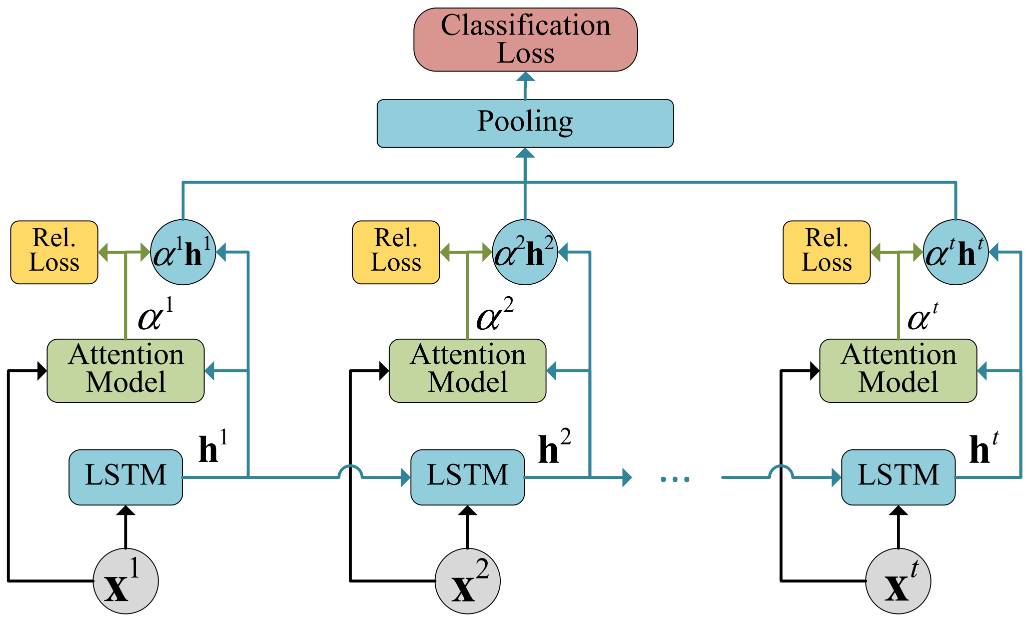

Figure 4.2: Demonstration of the proposed framework. $\mathbf{x}^t$ represents the feature of shot t. $\mathbf{h}^t$ corresponds to the temporal representation returned by LSTM at time t. α^t is the attention score for shot t, which is evaluated by the proposed attention model. Our framework jointly optimizes two objectives at two different levels. One is the relevance loss at shot-level, and the other is the classification loss at video-level.

Note that, for a description with multiple words, we use the average of these word vectors as its final representation. Now, for each event $e \in [1, E]$ we obtain a C-dimensional relevance score vector $\mathbf{S}^e \in \mathbb{R}^C$, in which each element indicates the relevance of the corresponding category to the target event e.

For a video v_i, we first segment it into a sequence of shots and sample one frame from each as its representation. For each shot t, a deep CNN [47, 88] pre-trained on ImageNet is used to output a 1000-way vector $\mathbf{p}_i^t$, which is a probability distribution over 1000 ImageNet categories. In [10], the final semantic relevance score of the t-th shot to the target event e is defined as the probabilistic expectation of the relevance scores over all 1000 categories.

$$r_i^{t,e} = \sum_{c=1}^{C} p_{i,c}^t S_c^e \tag{4.1}$$

where $p_{i,c}^t$ is the c-th element in the probability vector of the t-th shot in video v_i. However, the long tail of this distribution may pollute the final semantic relevance. Inspired by [36], we select the top 50 most responsive elements in $\mathbf{p}_i^t$ and re-normalize them with softmax. The expectation over this new distribution is taken as our final semantic relevance.

This type of semantic relevance is generated from both the image domain and the natural language domain. Semantic gaps certainly exist among language, image and video domains, resulting in low reliability compared with human-labelled semantic relevance, hence we call it weak semantic relevance. Note that this is only one method that can be used to calculate relevance and other methods such as the heuristic algorithm proposed in [89] can also be applied in our framework.

4.2.2 Problem Formulation

Suppose we have N labelled videos $(v_i, \boldsymbol{l}_i)$ in the training set, where $i \in [1, N]$, $\boldsymbol{l}_i \in \{0, 1\}^E$, l_i^e indicates whether v_i belongs to event e. The feature of the t-th shot from video v_i is represented as $\mathbf{x}_i^t$, where $t \in [1, M_i]$ and M_i is the total number of shots in v_i. Each video v_i is associated with a weak relevance vector $\mathbf{r}_i^e \in \mathbb{R}^{M_i}$, in which each element $r_i^{t,e}$ corresponds to the relevance score of shot $\mathbf{x}_i^t$ to the target event e. We denote the set of all videos and labels as V and L, respectively, and the set of relevance vectors of all videos as R. Under the guidance of the weak semantic relevance, the proposed attention model evaluates the attention score $\alpha_i^{t,e}$ for video shot $\mathbf{x}_i^t$ with regard to event e. Thus, the proposed framework pays a different amount of attention to different shots when conducting event classification.

To effectively leverage weak semantic relevance into our framework, we aim to maximize its utility with the attention model. To this end, we formulate the video event classification task assisted by weak semantic relevance by jointly optimizing the following two losses at two different levels, respectively:

$$Loss(V, L, R) = (1 - \lambda_a)L_c(V, L) + \lambda_a L_a(V, R) \tag{4.2}$$

where $L_c(V, L)$ is the classification loss corresponding to the groundtruth labels L, and $L_a(V, R)$ is the relevance loss at shot-level with respect to the guidance from weak semantic relevance R received by the attention model. λ_a is the parameter controlling the contribution of the guidance from the weak semantic relevance.

With Equation (4.2) as the objective function of the overall framework (illustrated in Figure 4.2), we develop the specific formulations of $L_c(V, L)$ and $L_a(V, R)$ in the following sections.

4.2.3 Video Event Classification by Paying Attention to Relevant Shots

It is natural to focus attention on relevant shots when performing event classification on a video. To achieve this, we use an attention score to measure the relevance of each video shot to its target event. The LSTM layer [31] in our framework is designed to capture the temporal information carried by the shots in a video. In each timestep, the LSTM unit returns the representation for the current shot, which memorizes useful patterns observed in its preceding video shots. We classify a video based on the representation sequence produced by the LSTM layer and the attention score assigned by the proposed attention model. The probability of video v_i being classified to the event e is denoted as p_i^e, which is formally defined as:

$$\begin{aligned} p_i^e &= f(\bar{\mathbf{h}}_i^e; \mathbf{w}_f) = \frac{\exp(\mathbf{w}_f^e \cdot \bar{\mathbf{h}}_i^e)}{\sum_{j\in[1,E]} \exp(\mathbf{w}_f^j \cdot \bar{\mathbf{h}}_i^j)} \\ \bar{\mathbf{h}}_i^e &= \frac{1}{Z_i^e} \sum_{t=1}^{M_i} \alpha_i^{t,e} \cdot \mathbf{h}_i^t \\ \mathbf{h}_i^t &= g_l(\mathbf{x}_i^t, \mathbf{h}_i^{t-1}; \mathbf{w}_l) \\ Z_i^e &= \sum_{t=1}^{M_i} \alpha_i^{t,e} \end{aligned} \tag{4.3}$$

$$\textit{where } t \in [1, M_i],\ e \in [1, E]$$

where $f(\cdot\,;\ \mathbf{w}_f)$ is the softmax scoring function, parameterized by $\mathbf{w}_f$. $[\mathbf{h}_i^1, \mathbf{h}_i^2, ..., \mathbf{h}_i^{M_i}]$ is the representation sequence produced by the LSTM layer, where $\mathbf{h}_i^t$ is the representation returned by the LSTM layer in timestep t. It is further weighted by the attention score sequence $[\alpha_i^{1,e}, \alpha_i^{2,e}, ..., \alpha_i^{M_i,e}]$ evaluated by the proposed attention model. The weighted average of this representation sequence, i.e. $\bar{\mathbf{h}}_i^e$, is taken as the input by the softmax function f. $g_l(\cdot\,,\ \cdot\,;\ \mathbf{w}_l)$ is the updating function within each LSTM unit and $\mathbf{w}_l$ is the corresponding parameters. The attention score $\alpha_i^{t,e}$ for shot $\mathbf{x}_i^t$ with regard to event e is calculated in each timestep by the attention model.

Accordingly, we define the video-level classification loss as the following categorical cross-entropy loss:

$$L_c(V, L) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{e=1}^{E} l_i^e \log(p_i^e) \tag{4.4}$$

Note that, for a video with multiple labels, we normalize its label vector $\boldsymbol{l}_i$ by L_1 norm to get a probability vector.

4.2.4 The Proposed Attention Model

The attention model [76] was recently incorporated into the LSTM framework for sequence generation tasks, such as image captioning [76, 114, 124] and video captioning [121]. The basic idea of it is that, when generating a caption for an image or video, in each timestep, the attention model computes the weight, i.e. attention score, for each individual visual region (e.g. image regions, video shots). Then, based on the combination of the weighted visual regions, the LSTM layer generates a word for the current timestep.

However, the above attention models are only supervised by the ground-truth labels, i.e. the captions of images or videos. To effectively leverage weak semantic relevance in video classification, we design a novel attention model which is not only supervised by the groundtruth event labels, but also guided by weak semantic relevance.

For a video v_i, in timestep t, we define the attention score vector $\boldsymbol{\alpha}_i^t$ for shot $\mathbf{x}_i^t$ by the following equations:

$$\begin{gathered}\boldsymbol{\alpha}_i^t = g_a(\mathbf{h}_i^t, \mathbf{x}_i^t; \mathbf{w}_a) \\ where\ t \in [1, M_i]\end{gathered} \tag{4.5}$$

where $g_a(\cdot\,,\,\cdot\,;\,\mathbf{w}_a)$ is an attention network with softmax output and being parameterized by $\mathbf{w}_a$. Each element $\alpha_i^{t,e}$ in $\boldsymbol{\alpha}_i^t$ is the attention score of shot $\mathbf{x}_i^t$ with respect to event e. We use a multi-layer perceptron as our attention network conditioned on shot feature $\mathbf{x}_i^t$ and its corresponding representation $\mathbf{h}_i^t$ produced by the LSTM layer.

Note that most existing attention models are designed for captioning, i.e. word sequence generation, where strong relationships exist between neighbouring words. Basically, these models compute the attention score for current timestep t, purely based on the previous representation $\mathbf{h}_i^{t-1}$ [114, 121]. In a video event classification task, we focus on the discriminative power of the final video representations. Therefore, our attention network is conditioned on $\mathbf{h}_i^t$ and $\mathbf{x}_i^t$. More specifically, we feed the concatenated vector $[\mathbf{h}_i^t, \mathbf{x}_i^t]$ to our attention network, where $\mathbf{h}_i$ captures the temporal information of the observed video shots and $\mathbf{x}_i$ preserves the inherent visual appearance of the current shot.

The weak semantic relevance cannot really be used as the attention score directly to weigh video shots, because it is noisy and not reliable enough. Therefore, instead of completely relying on it, we utilise it in our attention model as a weak guidance. The attention loss $L_a(V, R)$ is correspondingly formulated as:

$$L_a(V, R) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{\alpha}_i^t - \mathbf{r}_i^t\|^2 \tag{4.6}$$

where $\boldsymbol{\alpha}_i^t$ is the attention score vector of video v_i, calculated by Equation (4.5). This loss function implies that $\boldsymbol{\alpha}_i^t$ follows a Gaussian distribution with mean $\mathbf{r}_i^t$. Thus the proposed attention model takes the weak relevance as a priori when computing the attention score for the current shot.

The overall objective function, i.e. Equation (4.2), is optimized using stochastic gradient descent. By minimizing this objective function, our model exploits weak semantic relevance by the proposed attention model to facilitate video classification.

As emphasised before, the proposed attention model is supervised not only by video-level ground truth event labels, but also under the weak guidance of the shot-level semantic relevance. We can examine this by investigating the propagation path of the gradient with respect to attention scores: according to Equations (4.2), (4.3), (4.4), (4.5) and (4.6), the gradient with respect to the attention model is:

$$\frac{\partial Loss(V, L, R)}{\partial \boldsymbol{\alpha}_i} = (1 - \lambda_a) \frac{\partial L_c(V, L)}{\partial \bar{\mathbf{h}}_i} \cdot \frac{\partial \bar{\mathbf{h}}_i}{\partial \boldsymbol{\alpha}_i} + \lambda_a \frac{\partial L_a(V, R)}{\partial \boldsymbol{\alpha}_i} \tag{4.7}$$

Similarly, the LSTM layer is also supervised by these losses at two levels, respectively. The gradient with respect to the parameters of the LSTM layer, i.e. $\mathbf{w}_l$ is:

$$\begin{aligned} \frac{\partial Loss(V, L, R)}{\partial \mathbf{w}_l} &= (1 - \lambda_a) \frac{\partial L_c(V, L)}{\partial \bar{\mathbf{h}}_i} \cdot \frac{\partial \bar{\mathbf{h}}_i}{\partial \mathbf{w}_l} \\ &+ \lambda_a \frac{\partial L_a(V, R)}{\partial \boldsymbol{\alpha}_i} \cdot \frac{\partial \boldsymbol{\alpha}_i}{\partial \bar{\mathbf{h}}_i} \cdot \frac{\partial \bar{\mathbf{h}}_i}{\partial \mathbf{w}_l} \end{aligned} \tag{4.8}$$

The above equations clearly illustrate how the proposed framework learns from two different knowledge sources, i.e. event videos and Web-collected weak semantic relevance.

4.3 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of our framework and the ability of the proposed attention model to leverage weak semantic relevance.

4.3.1 Experiment Setup

Dataset. The performance study is conducted on three large-scale benchmark video event datasets, i.e. MEDTest14 [100], ActivityNet [30] and FCVID [43].

MEDTest14 [100] is a commonly-used benchmark dataset covering 20 events for complex video event classification. Each event has 100 positive training examples, and all events share about 5,000 negative training examples. The test set has approximately 23,000 videos.

ActivityNet [30] was recently released for complex human activity recognition. It comprises 28K of videos of 203 activity categories collected from YouTube. The video durations range from several minutes to half an hour and the total length of the whole dataset is 849 hours. Many of the videos in this dataset are shot by amateurs in uncontrolled environments, where the variances within the same activity category are often large. ActivityNet provides trimmed and untrimmed videos for evaluation. Following the settings in [112], we adopt a more challenging untrimmed setting for our experiments. ActivityNet consists of training, validation, and test splits. The test split is not publicly available, as the authors are reserving the test data for a potential future competition. Hence, we used the validation set as our test set as did [112].

FCVID [43] consists of 91,223 Web videos annotated manually into 239 categories. The total duration of all videos is 4,232 hours and the average duration per video is 167 seconds. The categories in FCVID cover a wide range of topics like social events (e.g. “tailgate party”), procedural events (e.g. “making cake”), objects (e.g. “panda”), scenes (e.g. “beach”), etc. We use its standard split of 45,611 videos for training and 45,612 videos for testing.

Implementation Details. Due to the computational limitations of our experimental environment, we construct a moderate sized network by segmenting each video into 30 shots. The colour histogram difference between consecutive frames is considered to be the indicator of shot boundaries. Other segmentation algorithms can also be employed in our framework. For videos with more than 30 shots, an agglomerative clustering alike method is applied to repeatedly merge two neighbouring shots whose total duration is the shortest into one in each round until total the number of shots is reduced to 30. For videos with less than 30 shots, we simply pad them with zeros at the tail. The middle frame of each shot is selected as its representative, and its feature is extracted by applying a very deep CNN architecture (from fc6 layer of VGG-19 [88]). We also use its probability output to compute the weak semantic relevance for each frame as explained in Section 4.2.1. Since unidirectional LSTM can only capture the previously observed temporal patterns (related to the current timestep) in a video, we adopt bidirectional LSTM [26, 86] to capture the intact temporal context (previous and post). We use the stochastic gradient descent algorithm with momentum to optimize our model. The batch size, momentum, and dropout rate (applied to both the LSTM layer and the fully connected layer) are set to 64, 0.9 and 0.1, respectively. The learning rate is set to 0.01 initially and divided by 10 after every 10K iterations. Finally, we employ mean average precision (mAP) to evaluate the overall performance on all three datasets.

Compared methods. The proposed approach is compared with the following alternative methods including two baseline methods and four state-of-the-art methods that also utilize weak semantic relevance generated from the Web:

1. SVM-WA. The weak semantic relevance is directly used to weigh video shot features without considering its reliability. The weighted shot features in a video are then average-pooled into a global feature vector, to which SVM is applied for classification.

2. LSTM-NR. It is a variant of the proposed method without utilizing weak semantic relevance. It is equivalent to LSTM with a conventional attention model.

3. Nearly-Isotonic SVM (NISVM) [10]. This state-of-the-art method sorts the video shots by their semantic relevance. An isotonic regularizer is introduced to impose larger weights on the shots with higher semantic relevance.

4. Ma *et al.* [71]. The authors downloaded 393K of event-related Web images from Google and Flickr by directly searching the event names. These Web images are assumed to be of high relevance to their corresponding events and are further used in fine-tuning CNNs.

5. Jiang *et al.* [43]. This method combines multiple state-of-the-art handcrafted visual features (e.g. improved dense trajectories) and deep features. The authors used a regularized deep neural network to exploit feature and class relationships.

6. OSF [112]. In this work, the CNNs were pre-trained on object and scene classification tasks and these were respectively applied to videos. The probabilistic outputs of these CNNs are

	ActivityNet	FCVID	MEDTest14
SVM-WA	50.8%	69.9%	28.1%
LSTM-NR	55.1%	73.2%	29.1%
Ours	**61.6%**	**77.8%**	**36.3%**

TABLE 4.1: Comparisons with baseline methods on the ActivityNet, FCVID and MEDTest14 datasets. SVM-WA only employs the semantic information as weights. LSTM-NR solely captures the temporal information and ignores the semantic information. Our full model jointly integrates the weak semantic relevance and the temporal information to facilitate video event classification.

considered to be the semantic relevance with respect to object and scene, respectively, and are used as the input features of a fusion network.

Although there are other video classification methods, they are either based on feature ensembles or a fusion of snippet scores [109] but do not utilize semantic information, hence they do not apply in our comparable experiments.

4.3.2 Comparison with Baseline Methods

To examine the extent to which the weak semantic relevance harvested from Web-knowledge can facilitate video classification, we compare our method with two baseline models SVM-MA and LSTM-NR. Table 4.1 shows the video classification performance of the evaluated baseline methods and the proposed approach. The proposed method outperforms SVM-MA by a large margin on each dataset, i.e. 10.8%, 7.9%, and 8.2% on ActivityNet, FCVID, and MEDTest14, respectively. This apparently supports our assumption discussed in Section 4.2.1 that utilising the weak semantic relevance without determining its reliability may result in inferior classification performance. The automatic learning process in our proposal effectively distinguishes useful information from noisy weak semantic relevance.

Our method is also compared with its variant LSTM-NR. The main difference between these two methods lies in the attention model training process, where the conventional attention model used in LSTM-NR is supervised by the groundtruth event label while our novel attention model also takes weak semantic relevance as a weak guidance. We get this variant by setting the parameter

	ActivityNet	FCVID
Ma *et al.* [71]	53.8%	-
Heilbron *et al.* [30]	42.5%	-
Jiang *et al.* [43]	-	73.0%
OSF [112]	56.8%	76.5%
Ours	**61.6%**	**77.8%**

Table 4.2: Comparisons with state-of-the-art methods on the ActivityNet and FCVID datasets. Our method achieve best classification performance on both of the two datasets.

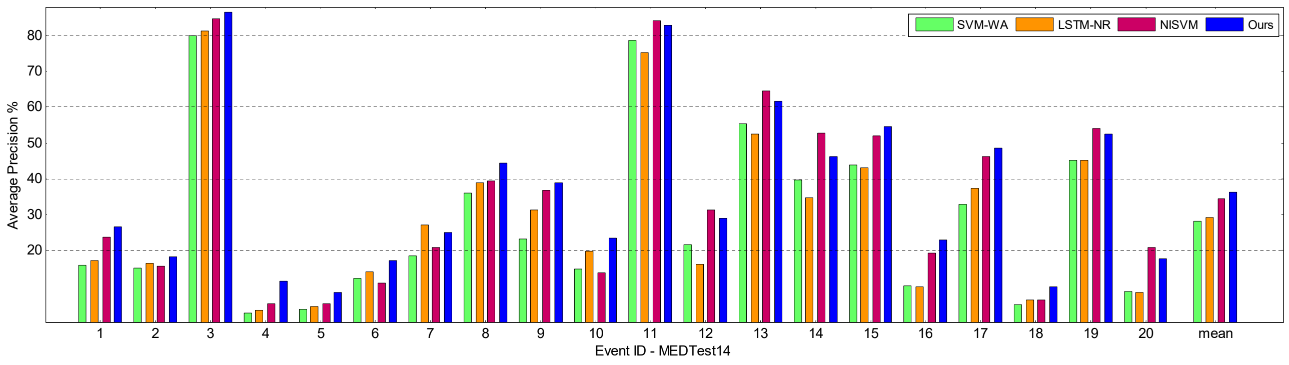

Figure 4.3: Results on the MEDTest14 dataset. The mean APs of SVM-WA, LSTM-NR, NISVM and our full model are 28.1%, 29.1%, 34.4% and 36.3%, respectively.

λ_a in Eq. (4.2) to 0. As shown in Table 4.1, the proposed model outperforms its variant for all three datasets. This indicates that our attention model which leverages semantic relevance as weak guidance is superior to the conventional model. The weak semantic relevance makes a significant contribution to achieving the promising classification performance.

4.3.3 Comparison with State-of-the-art Methods

In this section, we compare our method with four state-of-the-art methods: NISVM [10], Ma *et al.* [71], Jiang *et al.* [43], and OSF [112]. In Figure 4.3 and Table 4.2, we report the results of the performance study for all three datasets.

NISVM [10] is similar to our method in that both aim to assign larger weights to video shots with higher semantic relevance and the same sources to obtain weak semantic relevance are used. For a fair comparison, we adopt the same settings as used by [10]. On MEDTest14 dataset, we use Eq. (4.1) to compute the semantic relevance as in [10], without selecting the top 50 most responsive elements. We quote their best results to compare with ours. In Figure 4.3, the mean APs of NISVM and our model are 34.4% and 36.3%, respectively. Our method outperforms NISVM on 14 events out of 20 events. NISVM sorts the video shots by semantic relevance and only considers the ordering information among video shots. As discussed before, our method employs both the semantic relevance as a weak guidance to the proposed attention model and a bidirectional LSTM layer to capture the long-term temporal context among video shots. Hence, our model can exploit more valuable information from both of the semantic relevance and the temporal patterns in video shots.

For event categories 11, 12, 13 and 14, corresponding to "bee keeping", "wedding shower", "non-motorized vehicle repair", and "fixing musical instrument", our method does not perform as well as NISVM. After carefully investigating the videos for these four events, we find out that most of these videos are comprised of static scenes, such as "farm", and "church". As a result, the temporal information is overwhelmed by the strong static visual appearance and the LSTM layer in our model is overfitted. The fact that LSTM-NR performs even worse than SVM-WA on these four events also supports this observation.

In [71], Ma *et al.* evaluated several recently proposed very deep CNN architectures such as VGG-16, VGG-19 [88] and M2048 [11], for fine-tuning. For comparison, we took their best result on the ActivityNet dataset from their original paper. As seen in Table 4.2, the proposed method outperforms their method by a clear margin of 6.7% on ActivityNet. The possible reasons are as follows. Firstly, the compared method does not explicitly distinguish the reliability of the event-related images, which may introduce noise to the CNNs and be used for fine-tuning. It is not clear how robust the CNNs are to the noise. Secondly, an LSTM layer is used in our model to capture the temporal information in videos, while the CNNs used in [71] for fine-tuning can only capture the spatial visual appearance of images.

In Table 4.2 we present the best results from [112] on the ActivityNet and FCVID datasets from their original paper. This demonstrates the superior effectiveness of our model with regard to utilizing weak semantic relevance. Note that, their method leverages semantic relevance from three aspects i.e. object, scene, and low-level CNN feature, each of which corresponds to a different source domain. In this work, our method only utilizes one source of semantic relevance. However, it can be naturally

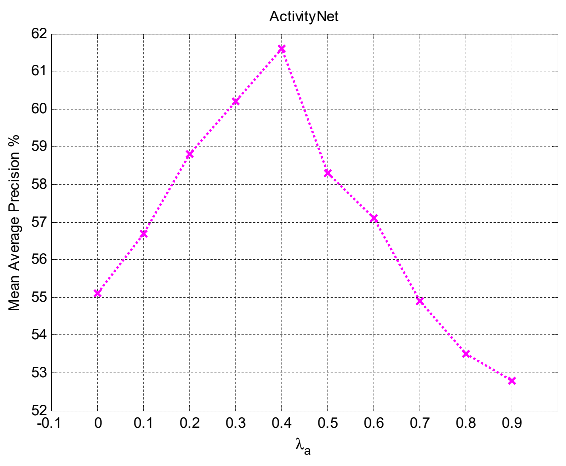

Figure 4.4: The effect of the relevance loss controlled by the trade-off parameter λ_a on the ActivityNet dataset.

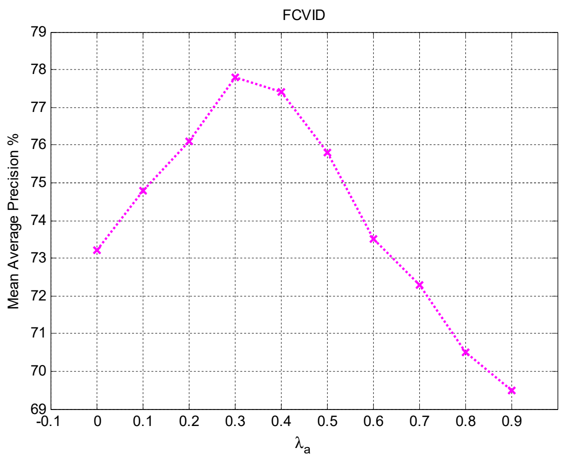

Figure 4.5: The effect of the relevance loss controlled by the trade-off parameter λ_a on the FCVID dataset.

extended to combine heterogeneous semantic relevance sources and is expected to achieve an even better performance.

Jiang *et al.* [43] combined multiple state-of-the-art handcrafted visual features (e.g. improved dense trajectories) and deep features for video event classification. They use a regularized deep neural network to exploit feature and class relationships. As clearly shown in Table 4.2, our model with the consideration of semantic relevance is more effective. In addition, we expect our method would be further improved by considering motion features for video shot representation, as for simplicity we only use static CNN features for our model in this work.

4.3.4 Experimental Study of The Contribution of Weak Semantic Relevance

In this section, we conduct an empirical analysis on the contribution of the weak semantic relevance. In Figure 4.4 and Figure 4.5 we compare the performance on the ActivityNet and FCVID datasets, respectively, of the proposed method using different values of the parameter λ_a in Eq. (4.2). A larger value of λ_a means a larger weight for the weak semantic relevance. On the ActivityNet dataset our model achieves the best classification performance when $\lambda_a = 0.4$, and on the FCVID dataset it works best when $\lambda_a = 0.3$. On both of these two datasets, when λ_a increases to more than 0.4, the classification performance drops dramatically which implies that when λ_a is greater than 0.4 our model starts to be dominated by the weak semantic relevance. This phenomenon can be understood as follows: the semantic relevance we extract from Web-knowledge is not reliable enough to contribute more than "40%" (corresponding to λ_a=0.4) to the classification task. If more reliable semantic relevance can be obtained, the value of the trade-off parameter should be increased, i.e. to allow semantic relevance contribute more for a better classification performance.

4.4 Summary

In this chapter, we propose a framework with a novel temporal attention model to automatically utilize weak semantic relevance to assist in the video classification task. This framework jointly optimizes two objectives at video-level and shot-level separately, which explicitly affect video classification at both global-level (i.e. video-level labels) and local-level (i.e. shot-level attention scores). To alleviate the effect of the noise introduced by the weak semantic relevance, we use weak semantic relevance as a weak guidance in the proposed temporal attention model, instead of considering it as the attention score directly. In this process, the LSTM layers model the temporal information to assist the attention model to generate semantic relevance score to weigh each shot. In return, the semantic relevance scores help the LSTM layers to generate compact temporal representation for each shot by filtering out noise. This process significantly improves the effectiveness of our proposed model. The semantic information and temporal information cooperate with each other again as discussed in Section 3.5.

Comprehensive performance studies have been conducted by comparing our method with six other methods over three large-scale benchmark datasets. The effectiveness of our method is exhibited by its superior performances compared with other models.

Our framework can also be smoothly extended and improved by generating weak semantic relevance from heterogenous information sources or combining multiple advanced visual features for video shot representation.

Chapter 5

Modelling Static Feature and Temporal Pattern for Video Event Retrieval

In the previous two chapters, we have studied how to collectively exploit semantic and temporal information for event classification. Besides classification, another import task of video event understanding is retrieval. In the following two chapters, we will focus on video event retrieval. Hashing-based methods are more and more popular for content-based visual retrieval due to the low storage cost and high retrieval efficiency. In this chapter, we will discuss how to integrate semantic (formalized as static visual feature in the following chapters) and temporal information for video hashing.

5.1 Introduction

Most of the existing work on video analysis generally resorts to pooling frame visual features into a single video representation, or pooling frame hash codes into a video hash code. For example, the method in [92], instead of pooling frame features, first generates a relaxed hash code for each frame. The relaxed hash codes of all frames in a video are then averaged into a single relaxed hash code to represent the entire video. Finally, the relaxed video code is binarized as the final video hash code. In these kind of hashing approaches, only the visual appearance, which is captured by static features, in each single frame is exploited, while the temporal patterns across the frame sequence are discarded.

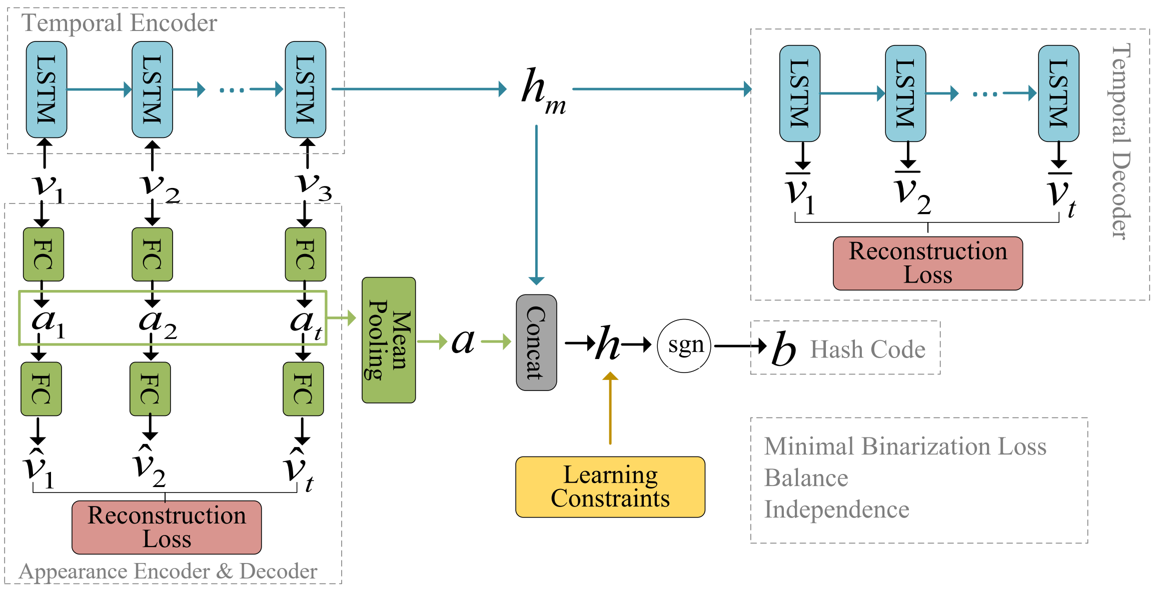

FIGURE 5.1: Illustration of the proposed model. The appearance encoder and decoder are comprised of fully connected layers (FC). The temporal encoder and decoder are built by LSTM layers. The two encoders are self-supervised by their own reconstruction loss, respectively. Given a video $[\boldsymbol{v}_1, ..., \boldsymbol{v}_2, \boldsymbol{v}_t]$, the appearance encoder outputs relaxed hash codes $[\boldsymbol{a}_1, ..., \boldsymbol{a}_2, \boldsymbol{a}_t]$ for all frames. Then we use mean pooling to aggregate them into a single representation $\boldsymbol{a}$ corresponding to the static visual appearance of a video. The temporal encoder generates relaxed hash code $\boldsymbol{h}_m$ corresponding to the temporal pattern. $\boldsymbol{h}_m$ and $\boldsymbol{a}$ are concatenated to $\boldsymbol{h}$. Three learning constraints with respect to hashing are imposed on $\boldsymbol{h}$ for jointly modelling the two encoders. Finally, $\boldsymbol{h}$ is binarized into hash code $\boldsymbol{b}$.

To utilize temporal patterns in video hashing, Zhang *et al.* [130] proposed the Binary Long Short-term Memory (BLSTM), which applies LSTM to capture the temporal information in videos. It outputs binary codes in each timestep with binarized LSTM cells. LSTM has outstanding performance when dealing with traditional sequential data, such as text and speech. In language corpus, there always exist strong relationships (e.g. long-term relationships between paragraphs or short-term relationships between words or sentences) that form informative temporal patterns. For the videos that record actions and sports, etc., their temporal patterns are obvious and are generally the dominant discriminative features for retrieval. Empirically, LSTM works well on these videos. However, due to the high diversity and complexity of video content [58], the temporal patterns in videos are not always helpful. For instance, the static visual appearance of scenic videos is certainly more discriminative for video retrieval compared to their temporal patterns. In this case, only considering the temporal patterns may lead to overfitting when learning the hash functions.

To solve the above dilemma, we propose to jointly model static visual appearance and temporal patterns for video hashing to maximise the benefits from both sides. To achieve this, we design two encoders to compose the hash function of our model, which are the temporal encoder and the appearance encoder, as shown in Figure 5.1. On one hand, they are designed to capture the temporal pattern at the video-level and the static visual appearance at the frame-level, respectively and they are self-supervised by their own reconstruction objectives. On the other hand, they are jointly learned under three hashing criteria: the minimal binarization loss, the balanced hash codes, and the independent hash codes. In this way, we can exploit both the temporal pattern and the visual appearance at the same time to collectively extract the most discriminative information from the video and accommodate effective video hashing.

The major contributions of our work are summarised as follows:

- In contrast to existing video hashing methods, which solely utilize the temporal patterns or the static visual appearance, we propose to collectively exploit these two aspects to facilitate effective video hashing. To the best of our knowledge, our model is the first unsupervised deep video hashing model that considers these two kinds of information at the same time.

- To jointly model the temporal patterns and the static visual appearance in a video, two encoders are jointly learned under three hashing criteria, i.e. the minimal binarization loss, the balanced hash codes, and the independent hash codes. They are imposed strictly and directly on the outputs of the two encoders without relaxation.

- We conduct extensive experiments on two large-scale video datasets, i.e. FCVID and ActivityNet. The experimental results demonstrate the effectiveness of the proposed video hashing model as state-of-the-art performance is achieved on both datasets.

5.2 The Proposed Model

In this section, we propose a novel framework for jointly modelling the static visual appearance and the temporal patterns for unsupervised video hashing. An overall problem formulation will be provided with detailed explanations of two deep encoders that capture the temporal patterns and the static visual appearance.

5.2.1 Problem Formulation

A video, consisting of a sequence of m frames, is represented as a matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m] \in \mathbb{R}^{d\times m}$, where $\boldsymbol{v}_t \in \mathbb{R}^d$ is the feature vector of the t-th frame. We aim to build a hash function $H : \mathbb{R}^{d\times m} \rightarrow \{-1, 1\}^k$ that can encode video $\boldsymbol{V}$ into a k-bit binary code $\boldsymbol{b} \in \{-1, 1\}^k$, where $k \ll d$. The generated binary code is expected to capture both visual and temporal information from the video.

Most of the existing video hashing methods only consider the static visual appearance, while the temporal nature of videos is neglected. For example, Song *et al.* [92] adopted a frame-level hashing scheme, which defines $H(V) = B(\frac{1}{m}\sum_{t=1}^{m} h(\boldsymbol{v}_t))$, where $h(\cdot)$ is a hash function on video frames and $B(\cdot)$ is a binarization operation predefined by the authors. It is obvious that the temporal information is discarded in $h(\cdot)$.

In order to capture the temporal patterns in videos, LSTM [31] is a natural choice. Given a video, LSTM generates a hidden representation $\boldsymbol{h}_t \in \mathbb{R}$ for the current timestep t, based on the current frame $\boldsymbol{v}_t$ and the hidden representation $\boldsymbol{h}_{t-1}$ of the timestep $t-1$:

$$\begin{aligned} \boldsymbol{h}_t &= f(\boldsymbol{v}_t, \boldsymbol{h}_{t-1}) \\ where \quad & t \in [1, m] \end{aligned} \tag{5.1}$$

where $f(\cdot,\cdot)$ is a non-linear updating function within LSTM. It aims to capture the informative temporal pattern embedded in the preceding $t-1$ frames and store it in a hidden representation $\boldsymbol{h}_{t-1}$. Recurrently, it produces the hidden representation $\boldsymbol{h_t}$ for frame sequence $[\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_t]$ based on $\boldsymbol{v}_t$ and $\boldsymbol{h}_{t-1}$.

To generate binary codes in video hashing, $f(\cdot,\cdot)$ can be modified for binarization as follows [130].

$$\begin{aligned} &\boldsymbol{b}_t = f(\boldsymbol{v}_t, \boldsymbol{b}_{t-1}) \\ &where \;\; \boldsymbol{b}_t \in \{-1, 1\}^k, \; t \in [1, m] \end{aligned} \tag{5.2}$$

As discussed previously, both the visual appearance and the temporal patterns have their own strengths and limitations in video hashing given the different types of videos. Therefore, instead of learning a video hash function solely based on one of the above features, we propose two encoders $H_T(\cdot)$ and $H_A(\cdot)$ corresponding to temporal patterns and static visual appearance respectively, based on which a joint hashing model is constructed.

$$H_T : \mathbb{R}^{d \times m} \to \{-1, 1\}^{k_t} \tag{5.3}$$

$$H_A : \mathbb{R}^{d \times m} \to \{-1, 1\}^{k_a} \tag{5.4}$$

In the following sections, we detail the formulations of two encoders and illustrate the framework of jointly modelling temporal patterns and static visual appearance for unsupervised video hashing.

5.2.2 The Temporal Encoder

For the temporal encoder $H_T(\cdot)$, we use LSTM to model the temporal patterns in videos. For video hashing, we need the final output of the encoder to be binary. Given that the vanilla LSTM [31] can only output real-value representations, one option for binarization is to use BLSTM as proposed in [130]. As demonstrated in Equation (5.2), BLSTM binarizes the hidden representation at each timestep. However, the target of our temporal encoder is to generate one final representation in binary code after observing the complete frame sequence of the video:

$$\boldsymbol{b}_T = H_T(\boldsymbol{V}) = H_T([\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m]) \tag{5.5}$$

Using binary intermediate hidden representations may cause information loss, and thus degenerates the capability of capturing temporal patterns. For this consideration, we only binarize the final representation returned back by LSTM. The proposed temporal encoder is formulated as follows:

$$\begin{aligned}
&\boldsymbol{b}_T = H_T(\boldsymbol{V}) = sgn(\boldsymbol{h}_m) \\
&\boldsymbol{h}_t = f(\boldsymbol{v}_t, \boldsymbol{h}_{t-1}) \\
&where \;\; t \in [1, m]
\end{aligned} \tag{5.6}$$

where $sgn(\cdot)$ is the sign function:

$$sgn(x) = \begin{cases} 1, & if \; x \geq 0 \\ -1, & otherwise \end{cases} \tag{5.7}$$

The detailed implementation of the LSTM updating function used in Equation (5.6) is listed below:

$$\boldsymbol{z}_t = \phi(\boldsymbol{W}_z\boldsymbol{v}_t + \boldsymbol{U}_z\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{5.8a}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i\boldsymbol{v}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{5.8b}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f\boldsymbol{v}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{5.8c}$$

$$\boldsymbol{c}_t = \boldsymbol{z}_t \circ \boldsymbol{i}_t + \boldsymbol{c}_{t-1} \circ \boldsymbol{f}_t \tag{5.8d}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o\boldsymbol{v}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{5.8e}$$

$$\boldsymbol{h}_t = \phi(\boldsymbol{c}_t) \circ \boldsymbol{o}_t \tag{5.8f}$$

$$\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5.8g}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.8h}$$

where $\boldsymbol{W}$ are the input weight matrices, $\boldsymbol{U}$ are the recurrent weight matrices, and $\boldsymbol{b}$ are the bias vectors. $\sigma(\cdot)$ is the *sigmoid* recurrent gate activation function. $\phi(\cdot)$ is the input activation function. Both $\sigma(\cdot)$ and $\phi(\cdot)$ are element-wise functions. For $\phi(\cdot)$, we use the *tanh* function. Some other non-linear functions can also be employed as the input activation functions. Since our binary codes are modelled to be $\{-1, 1\}$, $\phi(x) = tanh(x) \in (-1, 1)$ is a natural choice for the hashing task.

5.2.3 The Appearance Encoder

The appearance encoder focuses on exploiting the static visual appearance in video frames. Motivated by the recent advances achieved by deep hashing methods on images [17,50,60,65], we design a deep encoding network as the building block of our appearance encoder. It works on video frames and is comprised of multiple stacked layers of non-linear transformations. By providing a video frame $\boldsymbol{v}_t$ to the encoding network, it will generate a relaxed hash code $\boldsymbol{a}_t \in (-1, 1)^{k_a}$.

Assume there are L layers in our encoding network and the l-th layer contains d^l units, where $l \in [1, L]$. The formulation of encoding network is as follows:

$$\begin{aligned} \boldsymbol{a}_t &= g(\boldsymbol{v}_t) = \psi(\boldsymbol{W}^L \boldsymbol{x}_t^{L-1} + \boldsymbol{c}^L) && (5.9a) \\ \boldsymbol{x}_t^l &= \psi(\boldsymbol{W}^l \boldsymbol{x}_t^{l-1} + \boldsymbol{c}^l) && (5.9b) \\ \boldsymbol{x}_t^0 &= \boldsymbol{v}_t && (5.9c) \\ &\textit{where } l \in [1, L] \end{aligned}$$

where $\boldsymbol{W}^l$ are the projection matrices in the l-th layer, $\boldsymbol{b}^l$ are the bias vectors at the l-th layer, and $\psi(\cdot)$ is the *tanh* activation function.

Now we have the building block of the appearance encoder. In contrast to the conventional encoding networks, which take a single image as the input, our appearance encoder is expected to cope with videos consisting of a sequence of frames. A straightforward approach is to pool all frames in a video into one single video representation; however, information loss may occur in the pooling procedure. To exploit every detail in each single frame, we first generate a relaxed hash code $\boldsymbol{a}_t$ for each frame $\boldsymbol{v}_t$ in a video and binarize the average of the relaxed hash codes of all frames as the final hash code for the video. This strategy is also employed in [92]. From the point of view of training deep neural networks, training the encoding network on frames takes advantage of the abundant information to be found at the frame-level. The appearance encoder is designed as follows:

$$\begin{aligned} \boldsymbol{b}_A &= H_A(\boldsymbol{V}) = H_A([\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m]) \\ H_A(\boldsymbol{V}) &= sgn(\boldsymbol{a}) \\ \boldsymbol{a} &= \frac{1}{M} \sum_{t=1}^{M} \boldsymbol{a}_t \end{aligned} \tag{5.10}$$

Since we already have the temporal encoder to manage the temporal patterns, the appearance encoder can focus on exploiting the static visual appearance in each frame for video hashing. In the next section, we will discuss how the proposed joint model collectively learns these two encoders.

5.3 Jointly Learning The Temporal Encoder and Appearance Encoder

In this section, we will discuss the learning objectives of the proposed temporal encoder and appearance encoder in a self-supervision scenario and explain how to jointly model temporal patterns and static visual appearance for video hashing.

5.3.1 Self-supervised Learning

We employ the self-supervised encoder-decoder framework [46, 93, 95, 103, 130] to train the two proposed encoders. There are many successful applications of the encoder-decoder framework, such as image representation learning [46,103], language modelling learning [95] and video representation learning [93]. In this work, we utilize this framework for unsupervised video hashing.

For the temporal encoder, we use the temporal order of the video illustrated by a sequence of frames as a self-supervision. We assume that, if the hash code of a video produced by the temporal encoder has captured sufficient informative temporal patterns from the video, the hash code can be decoded to the original sequence of frames by the temporal decoder. The temporal decoder $\bar{H}_T : \mathbb{R}^{k_t} \to \mathbb{R}^{d \times m}$ is defined as:

$$[\bar{\boldsymbol{v}}_1, \bar{\boldsymbol{v}}_2, ..., \bar{\boldsymbol{v}}_m] = \bar{H}_T(\boldsymbol{h}_m) \tag{5.11a}$$

$$\bar{\boldsymbol{v}}_t = \bar{\boldsymbol{W}}\bar{\boldsymbol{h}}_t + \bar{\boldsymbol{c}} \tag{5.11b}$$

$$\bar{\boldsymbol{h}}_t = \bar{f}(\bar{\boldsymbol{h}}_{t-1}, \boldsymbol{h}_m) \tag{5.11c}$$

$$where \ \ t \in [1, m] \tag{5.11d}$$

Here, we also use LSTM as our temporal decoder. $\bar{f}(\cdot)$ is the updating function used by the decoding LSTM and $\bar{\boldsymbol{h}}_t$ is the hidden representation returned by the decoding LSTM at timestep t. In particular, since $\bar{\boldsymbol{h}}_t \in (-1, 1)^{k_t}$, we employ a linear transformation to reconstruct $\bar{\boldsymbol{h}}_t$ to the original space where the frame feature $\boldsymbol{v}_t$ lives. Ideally, we could reconstruct the video from the binary hash code $\boldsymbol{b}_T$. However, because the derivative of the *sgn* function is zero in almost every case, we resort to reconstructing the video from $\boldsymbol{h}_m$. Later, we will introduce a binarization loss to alleviate the information loss caused by the *sgn* function.

Based on the reconstruction of the temporal decoder, given a video, the learning objective of the temporal encoder can be formulated as follows:

$$L_T = \frac{1}{M}\sum_{t=1}^{M} \|\bar{\boldsymbol{v}}_t - \boldsymbol{v}_t\|_2^2 \tag{5.12}$$

For the appearance encoder, we also develop a deep decoding network. Similarly, the decoding network reconstructs each frame from the relaxed hash code $\boldsymbol{a}_t$. The appearance decoder $\bar{H}_A : \mathbb{R}^{k_a} \rightarrow \mathbb{R}^d$ is as follows:

$$\hat{\boldsymbol{v}}_t = \hat{g}(\boldsymbol{a}_t) = \psi(\hat{\boldsymbol{W}}^L \hat{\boldsymbol{x}}_t^{L-1} + \hat{\boldsymbol{c}}^L) \tag{5.13a}$$

$$\hat{\boldsymbol{x}}_t^l = \psi(\hat{\boldsymbol{W}}^l \hat{\boldsymbol{x}}_t^{l-1} + \hat{\boldsymbol{c}}^l) \tag{5.13b}$$

$$\hat{\boldsymbol{x}}_t^0 = \boldsymbol{a}_t \tag{5.13c}$$

$$where\ l \in [1, L]$$

Similarly, the learning objective of the appearance encoder can be formulated as follows:

$$L_A = \frac{1}{M}\sum_{t=1}^{M} \|\hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\|_2^2 \tag{5.14}$$

5.3.2 Jointly Modelling

The temporal encoder and the appearance encoder focus on managing the temporal patterns and the static visual appearance of a video, respectively. Recall that, given a video, the temporal encoder outputs hash code $\boldsymbol{b}_T \in \{-1, 1\}^{k_t}$ and the appearance encoder produces hash code $\boldsymbol{b}_A \in \{-1, 1\}^{k_a}$. To jointly model these two types of information in a video, we propose to concatenate these two hash codes to get the final hash code $\boldsymbol{b} \in \{-1, 1\}^k$, where $k = k_t + k_a$. Three learning constraints are imposed for jointly modelling the two encoders and generating high quality hash code $\boldsymbol{b}$.

We formulate the aforementioned encoders and decoder as a deep neural network and the training of a deep neural network relies on gradient back propagation. Due to the fact that the derivative of the *sgn* function is zero in almost every case, we resort to working on the relaxed hash code before the *sgn* function and define it as a concatenation of $\boldsymbol{h}_m$ and $\boldsymbol{a}$:

$$\boldsymbol{h} = (\boldsymbol{h}_m, \boldsymbol{a}), \ \ \boldsymbol{h} \in (-1, 1)^k \tag{5.15}$$

The final hash code $\boldsymbol{b}$ is defined as:

$$\boldsymbol{b} = sgn(\boldsymbol{h}), \ \ \boldsymbol{b} \in \{-1, 1\}^k \tag{5.16}$$

Given a video, to reduce the information loss caused by the *sgn* function, we impose the first learning constraint on $\boldsymbol{h}$:

$$L_{bin} = ||\boldsymbol{b} - \boldsymbol{h}||_2^2 \tag{5.17}$$

To produce a high quality hash code [110], we require that each bit in the hash code has a 50% chance of being -1 or 1 (balance criterion), and different bits are independent to each other (independence criterion). Suppose there are N videos in the training set, then $\boldsymbol{h}^i$ and $\boldsymbol{b}^i$ are the relaxed code and binary code of the i-th video, respectively. The learning constraint corresponding to the balance criterion can be formulated as:

$$L_{bal} = ||\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}^i||_2^2 \tag{5.18}$$

The learning constraint with respect to the independence criterion is defined as:

$$\begin{aligned} L_{indep} &= ||\frac{1}{N}\boldsymbol{H}\boldsymbol{H}^T||_2^2 \\ \boldsymbol{H} &= [\boldsymbol{h}^1, \boldsymbol{h}^2, ..., \boldsymbol{h}^N] \end{aligned} \tag{5.19}$$

where, $\boldsymbol{H} \in (-1, 1)^{k \times N}$ is a matrix. Its i-th column is the relaxed code $\boldsymbol{h}^i$ of the i-th video.

The hash code $\boldsymbol{b}$ consists of two parts of code generated from two different encoders. These two encoders are self-supervised separately by their own reconstruction losses, i.e. Equation (5.12) and Equation (5.14). Under the above three learning constraints, the two separated encoders are unified for jointly modelling temporal patterns and visual appearance in a video. The independence criterion forces different bits to be uncorrelated and furthermore promotes the two parts of the hash code, generated from the two encoders, to be uncorrelated. This is beneficial to our hashing model with regard to learning more powerful hash codes by reducing the redundant information captured by both of the temporal encoder and the appearance encoder.

In summary, the whole learning objective can be formally defined as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}(L_T^i + L_A^i + \lambda_1 L_{bin}^i) + \lambda_2 L_{bal} + \lambda_3 L_{indep} \tag{5.20}$$

where L^i is the loss function corresponding to i-th video.

5.3.3 Architecture Details

We use one LSTM layer as our temporal encoder to produce a k_t-dimensional hash code. For the appearance encoder, we use three fully connected layers with $d/2$, $2k_a$ and k_a units, respectively, and it generates a k_t-dimensional hash code. For simplicity, we set $k_t = k_a = k/2$. For the temporal decoder, we use one LSTM layer followed by a fully connected layer with linear activation. The decoding LSTM has $k_t/2$ units, and the following linear fully connected layer projects the $\frac{k_t}{2}$-dimensional output of the decoding LSTM to a d-dimensional vector as a reconstruction. There are also three fully connected layers in the appearance decoder, which have $2k_a$, $d/2$, and d units, respectively. The whole framework first compresses high-dimensional video features to low-dimensional hash codes in the encoding stage, and reconstructs the video features from the hash codes in the decoding stage. The hash code typically holds much less dimensions than the original video feature and in order to minimize the information loss in the encoding stage, the dimensionality of the video feature is reduced gradually. Therefore, the unit numbers of each layer in the appearance encoder are set to $d/2$, $2k_a$ and k_a and accordingly, the unit numbers of each layer in the appearance decoder are set to $2k_a$, $d/2$, and d. It is possible to use more LSTM layers with reducing unit numbers for the temporal encoder; however, it has been observed that the performance improvement achieved by more LSTM layers is limited and it also requires longer training time. Hence, only one LSTM layer for encoding and another one for decoding are used in the proposed model. Note that, this work is focused on how to jointly exploit visual appearance and temporal patterns for unsupervised video hashing. It is straightforward for our framework to incorporate different architectures for each individual encoder and decoder.

5.4 Experiments

In this section, we conduct extensive experiments on two large-scale datasets to evaluate the effectiveness of the proposed unsupervised video hashing model.

5.4.1 Experimental Settings

We evaluate the proposed model on two challenging large-scale video datasets, i.e. ActivityNet [30] and FCVID [43].

ActivityNet [30]. This recently released video dataset covers a wide range of complex human activities that are of interest to people in their daily living. It comprises 28K of videos of 203 activity categories collected from YouTube. The lengths of the videos range from several minutes to half an hour. The total length of the whole dataset is 849 hours. Many of the videos in this dataset are shot by amateurs in uncontrolled environments, where the variances within the same activity category are often large. ActivityNet provides trimmed and untrimmed videos for evaluation. Here we adopt the more challenging untrimmed videos for our experiments.

FCVID. Fudan-Columbia Video Dataset [43] consists of 91,223 Web videos annotated manually into 239 categories. The total duration of all videos is 4,232 hours and the average duration per video is 167 seconds. The categories in FCVID cover a wide range of topics like social events (e.g. "tailgate party"), procedural events (e.g. "making cake"), objects (e.g. "panda"), scenes (e.g. "beach"), etc. We use its standard split of 45,611 videos for training the proposed video hashing model and 45,612 videos for retrieval.

Evaluation Protocols. We employ Average Precision at top K retrieved videos (AP@K) for retrieval performance evaluation. For a given query, the definition of AP@K is as follows:

$$\begin{aligned} AP@K = \frac{1}{min(R,K)} \sum_{i=1}^{K} \frac{R_i}{i} \times I_i, \\ where \quad 1 \leq i \leq K, \end{aligned} \tag{5.21}$$

we rank the retrieved videos by their hamming distances to the query. R_i is the number of relevant videos in the top i videos in the ranking list. R is the number of relevant videos in the database. $I_i = 1$ if the i-th video is relevant and 0 otherwise. Two videos are considered to be relevant if they belong to the same class. For each class, we randomly sample 10 videos as queries. As a result, we get about 2000 queries for each dataset. Then the mean of AP@K of the queries is used as a performance metric for each dataset. For the FCVID dataset, we sample queries from the test set, and the rest of the test set is used as the database. For the ActivityNet dataset, the labels of the test set are not publicly available, as the authors are reserving the test data for a potential future competition. Hence, we sample queries from the validation set, and the rest of the validation set and training set is used as the database. We evaluate code length $k \in [16, 32, 64, 128, 256]$.

Implementation Details. Our unsupervised video hashing model comprises two LSTM layers (temporal encoder and decoder), and seven fully connected layers (one layer for temporal decoder, and six for appearance encoder and decoder). As a compromise for training time and GPU memory, we uniformly sample 30 frames for each video. In this setting, our model is a deep framework containing 67 layers after unrolling the two LSTM layers. We believe that a stronger model can be achieved by increasing the frame sampling rate. For each frame, we employ ResNet [29] and extract a 2048-D CNN feature as its representation. To achieve a fair comparison, the ResNet feature is used for all compared methods. The stochastic gradient descent algorithm with momentum is used to train our model and the batch size, momentum, and dropout rate (applied on both the LSTM layer and the fully connected layer) are set to 150, 0.9 and 0.1, respectively. The learning rate is set to 0.01 initially and divided by 5 after every 10K iterations.

The three parameters λ_1, λ_2 and λ_3 are set to 0.1, 1 and 1, respectively by cross validation and it is observed that the proposed model is not sensitive to these parameters. In this work, we assign equal weights to L_T and L_A. How to adaptively assign weights to L_T and L_A will be studied as a future work.

Compared methods. To validate the effectiveness of the proposed model (JTAE), we first conduct experimental studies by comparing it with three baseline models. Furthermore, we compare the performance of our model with three state-of-the-art unsupervised video hashing methods.

1. AE. Appearance encoder only. We only use the appearance encoder to produce the hash code for a video and this baseline model discards all the temporal patterns in the videos. We can implement this model by removing the loss function L_T in Equation (5.20), which corresponds to the temporal encoder.

2. TE. Temporal encoder only. Similar to AE, this variant only focuses on the temporal patterns in videos. This model can be obtained by deleting the loss function L_A in Equation (5.20), which

corresponds to the appearance encoder.

3. TAE. Temporal encoder plus appearance encoder. To evaluate to what extent the joint modelling of the temporal encoder and the appearance encoder can improve the quality of the learned hash code, we develop this baseline model by simply combining these two encoders and discarding the three learning criteria. It is implemented by removing the following three terms, i.e. L_{bin}, L_{bal}, L_{indep} in Equation (5.20).

4. ITQ. This is a very popular unsupervised hashing method called Iterative Quantization [25]. It first employs average pooling on frames in a video to get a video-level representation. Then PCA is used to reduce the dimensionality of the video representation to the target code length (i.e. k). Finally, it iteratively learns a rotation which minimizes the quantization loss.

5. MFH. This is a multi-feature hashing method that works at frame-level [92] and exploits the similarity graph of the frames. It first obtains the mean of the relaxed hash codes of frames in a video, then binarizes the mean as the final hash code. This method discards the temporal information in videos.

6. BLSTM. This is an unsupervised video hashing method recently proposed by Zhang *et al.* [130]. This method utilizes the temporal patterns in videos. Given a video, it recurrently outputs binary codes at each timestep using binary LSTM.

5.4.2 Comparison with Baseline Methods

Table 5.1: Effects of jointly modelling

mAP@20		k=256	k=128	k=64	k=32	k=16
FCVID	JTAE	**0.307**	**0.263**	**0.201**	**0.133**	**0.061**
	TAE	0.276	0.237	0.174	0.107	0.052
ActivityNET	JTAE	**0.188**	**0.150**	**0.104**	**0.055**	**0.026**
	TAE	0.166	0.135	0.086	0.047	0.019

Our model aims to exploit both the temporal patterns and static visual appearance for video hashing. To evaluate to what extent these two kinds of information can contribute to the hashing performance, we design three variants, i.e. AE, TE and TAE, of the proposed model to compare against. First, in Figure 5.2 and Figure 5.3 we report the hashing performance of these methods when using 128-bit and 256-bit hash codes. On the FCVID dataset, TE outperforms AE which indicates the temporal patterns are more discriminative than the visual appearance for most of videos is this dataset. On the contrary, AE outperforms TE on the ActivityNet dataset which indicates the visual appearance is more important for this dataset.

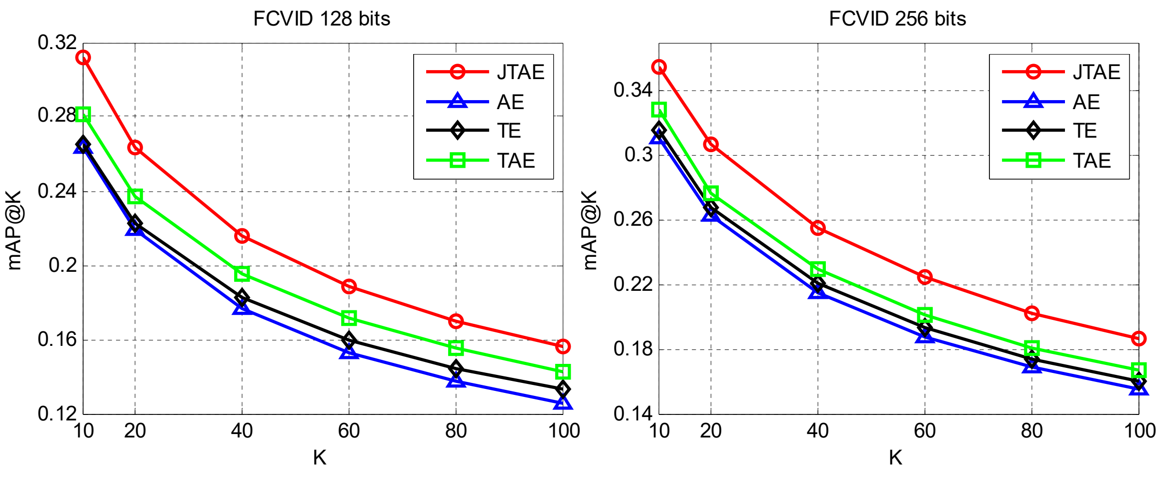

Figure 5.2: Comparison with baseline methods on the FCVID dataset using 128-bit and 256-bit hash codes

For both datasets, we can see TAE outperforms TE and AE which supports our proposal that combining temporal patterns and visual appearance can effectively facilitate video hashing. However, the performance improvement is limited. We analyze the reason as follows. Visual appearance exists in each frame, and temporal patterns are captured using LSTM by watching each frame. In other words, temporal patterns are summarized from the visual appearance of frames. Although a temporal pattern can be viewed as a high-level encoding of static visual appearance, the simple concatenated hash code (half bits from temporal encoder, and half bits from appearance encoder) of TAE captures redundant information. Hence, we propose to jointly model these two encoders by imposing three criteria (i.e. minimal binarization loss, balance and independence) on the combined hash code. As shown in Figure 5.2 and Figure 5.3, the proposed model JTAE outperforms TAE by clear margins on

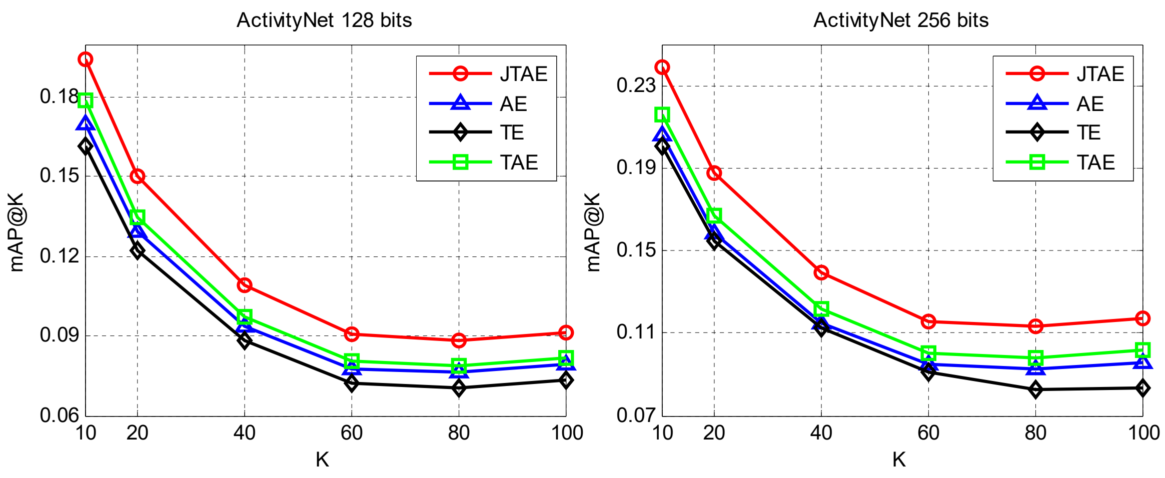

Figure 5.3: Comparison with baseline methods on the ActivityNet dataset using 128-bit and 256-bit hash codes

both of the datasets. In Table 5.1 we also report mAP@20 of JTAE and TAE when using bit lengths of 256, 128, 64, 32, and 16, respectively. Again, JTAE outperforms TAE consistently on both datasets. The above experimental results prove the effectiveness of the proposed unsupervised hashing model of jointly modelling temporal patterns and visual appearance.

5.4.3 Comparison with State-of-the-art Methods

In this section, we compare the proposed model with three state-of-the-art hashing methods, i.e. ITQ [25], MFH [92] and BSLTM [130] and we report the experimental results on the FCVID datset and the ActivityNet dataset in Figure 5.4 and Figure 5.5, respectively.

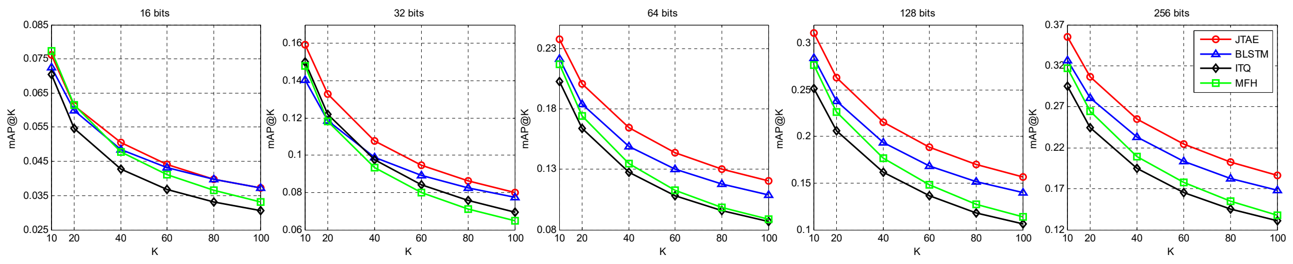

Figure 5.4: Comparison with state-of-the-art methods on the FCVID dataset.

ITQ and MFH are methods that only consider static visual appearance. ITQ first pools the frame features from a video into a single video-level feature, then performs hashing based on the video-level

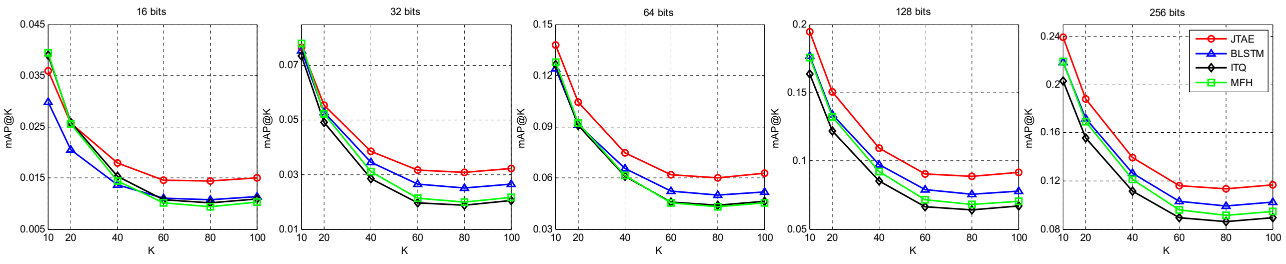

Figure 5.5: Comparison with state-of-the-art methods on the ActivityNet dataset.

features. MFH conducts hashing on frames first and binarizes the mean of real-valued frame-level hash codes as the final video hash code. BLSTM focuses on exploiting temporal patterns in videos. In contrast to vanilla LSTM [31], BLSTM adopts binarized hidden representation instead of real-values. Hence, it can directly produce hash codes in each timestep without binarization loss. As shown in Figure 5.4 and Figure 5.5, on the FCVID dataset, BLSTM performs better than ITQ and MFH when using longer hash codes (i.e. when $k = 64, 128, 256$). And on the ActivityNet dataset, though BLSTM still outperforms ITQ and MFH, the gap is not as large as on the FCVID dataset. This observation is consistent with the phenomenon discussed in Section 5.4.2, that the temporal information is more important than the visual appearance on the FCVID dataset and vice versa on the ActivityNet dataset.

On different datasets, although the temporal pattern and static visual appearance are not equally important, we can not discard the weaker one. To maximize the utility of these two kinds of information, we propose to simultaneously exploit the temporal patterns and visual appearance. Our model outperforms all other state-of-the-art methods on both of the two datasets. Compared to the methods which only utilize visual appearance, i.e. ITQ and MFH, our method has superior performance by taking extra temporal information into consideration. Compared to BLSTM, besides exploiting visual appearance, our model strictly imposes the balance and independence criteria on our two encoders, where BLSTM approximates these two criteria by using batch normalization [33] on the cell state in the LSTM layer.

We also note that when use shorter bit length (e.g. 16), BLSTM and our model both perform poorly compared to other methods. The same point we share is that, we both generate hash code by using the hidden representation of LSTM. The bit length of the hash code is same as the dimension of the hidden representation. A possible reason is that LSTM needs a higher dimensional hidden representation to perform better for memorizing the complex temporal patterns in videos. Many existing applications of LSTM [27, 93, 95, 104] have shown that a higher dimensionality of hidden representation leads to a better performance.

5.5 Summary

In this chapter, we propose a novel unsupervised video hashing model. In contrast to existing conventional hash methods that only utilize static visual appearance (corresponding to semantic information) or solely focus on temporal patterns, the proposed model exploits these two types of information by collectively learning two encoders, i.e. the temporal encoder and the appearance encoder. Our model maximizes the utility of temporal patterns and semantic information by imposing three learning criteria on the two encoders directly and strictly. In this way, the two kinds of information are jointly exploited and the redundant information captured by the two encoders can be reduced, hence high quality hash codes can be generated. To the best of our knowledge, our proposal is the first unsupervised deep video hashing model that can exploit temporal pattens and semantic information simultaneously.

Comprehensive experiments have been conducted on two challenging large-scale video datasets, FCVID and ActivityNet. In addition, the effectiveness of the proposed method was verified by comparisons with six alternative video hashing methods.

Essentially, our model is an end-to-end framework which can be easily extended for multi-view or multi-modal hashing. Moving on from this, query and search between data in different formats is a potential future research direction. For example, we can search videos by photos shot by mobile devices. Furthermore, if considering label information, our model can also be extended to supervised or semi-supervised hashing framework.

Chapter 6

Video Retrieval via Adaptive Selection

In the previous chapter, we have explored how to jointly utilize the semantic and temporal information by two concatenated encoders and manually imposed learning criteria for video hashing. The two encoders are separated without interaction and the learning criteria only works on loss-level. To further study how to adaptively exploit these two kinds of information, in this chapter we propose a novel Adaptive Selection mechanism which enables the two types of information to interact and cooperate with each other. Thus, the complementary advantages of semantic information and temporal pattern can be utilized more effectively.

6.1 Introduction

Unlike the flourishing domain of image retrieval [15, 66, 87, 91, 107, 117], video hashing has not been studied thoroughly due to the challenging temporal nature of videos. However, thanks to rapid advances in mobile video capturing devices and network connections, more and more users prefer to use videos to record their daily life rather than photos. The quantity of video content is exploding on the Web (e.g. YouTube, Snapchat, and Twitch). Therefore, advanced hashing techniques [92,126,130] for large-scale video retrieval are in high demand.

In contrast to static images, a video consists of a sequence of frames where inherent temporal patterns exist. For example, a short-term temporal pattern may lie in an object's motion or a human action, and an event [57,58] lasting ten minutes may contain long-term temporal information. A video is far beyond simply a set of images. Owing to the temporal nature plus the diverse and complex visual appearance of each frame, video retrieval is much more challenging than plain image retrieval.

Although the aforementioned learning to hash methods perform well on image retrieval, they cannot be smoothly transplanted to a video retrieval task as they are not capable of utilizing the underlying temporal patterns. Recently, a few researchers have attempted to employ the underlying temporal information for video hashing with deep learning techniques. Zhang *et al.* [130] focused on modelling discriminative temporal patterns with Long Short-term Memory (LSTM). LSTM has shown its superior performance in modelling sequential data such as text and speech, as in language, there exist plentiful strong relations (words to words in a sentence, sentence to sentence in a paragraph) which form informative temporal patterns. However, this is not always true in videos due to the high diversity and complexity of video content [57, 58]. For instance, discriminative temporal patterns can be extracted from videos recording actions and sports, while in scenic videos, static visual features are typically more discriminative. Although, in Chapter 5, we proposed a self-supervised hashing method, where temporal pattern and static visual features are jointly modelled, the contributions from each aspect need to be further investigated .

In summary, the powerful representative abilities inherent in both temporal patterns and static visual features have not been effectively utilized in the existing video hashing methods. It has been proven that these two aspects are complementary to each other as a pair of partners that can supply more comprehensive information [58, 59] for video hashing; however, existing methods lack optimal feature integration for video representation. To address these problems, we propose a dual-stream deep framework to adaptively model static visual features and temporal patterns. In each stream, the most important components of the temporal patterns (or the static visual features) are selected by taking into account the complementary information carried by its partner static visual feature (or temporal patterns), based on which a finer feature representation is generated. To achieve that, a novel information filtering mechanism is designed, which is called Adaptive Selection (AS). An intermediate video representation is generated in the form of an optimal integration of refined temporal patterns and static visual features, based on which hash codes are finally produced. The main contributions of our work are summarized as follows:

- In contrast to most of the existing video hashing methods, which solely concentrate on temporal patterns or static visual features, we propose to adaptively select complementary information from each aspect to facilitate video hashing.

- To adaptively model the temporal patterns and the static visual features in videos, we propose a dual-stream network equipped with a novel Adaptive Selection mechanism. One stream is dedicated to static visual features and the other to modelling temporal patterns. The proposed AS mechanism consults an auxiliary conditional input to select components from the main input, which are complementary to the auxiliary input. The AS enables our framework to select informative components from both aspects adaptively and to complementarily integrate them to generate high quality hash codes.
- We conduct extensive experiments on two large-scale benchmark video datasets, i.e. FCVID and ActivityNet. The significant performance improvement upon the existing methods verifies the effectiveness of the proposed video hashing framework.

6.2 The Proposed Method

In this section, we design a deep learning to hash framework for video retrieval. Using a dual-stream architecture, it adaptively exploits complementary advantages from two essential aspects of a video, i.e. static visual features and temporal patterns. To this end, we propose a novel Adaptive Selection mechanism, which enables the dual-stream framework to adaptively integrate complementary components (for each aspect respectively) from the original representations to generate high quality hash codes.

6.2.1 Problem Formulation

In the training dataset, we have n videos $\mathcal{V} = \{\boldsymbol{V}_i\}_{i=1}^{n}$, each with one or more semantic labels. A video is comprised of a sequence of m frames. We can describe it as a matrix $\boldsymbol{V} = [\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^m] \in \mathbb{R}^{d\times m}$, where $\boldsymbol{x}^t \in \mathbb{R}^d$ is the d-dimensional feature vector of the t-th frame. In this work, we focus on video hashing which can preserve semantic similarity between videos. If $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ are similar, their similarity label s_{ij} is set to be 1, otherwise $s_{ij} = 0$. The set of all similarity labels is denoted as $\mathcal{S}$ and it can be constructed from semantic labels. For instance, s_{ij} can be set to 1 if $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ are assigned to the same semantic label. We aim to build a non-linear hash function $f : \mathbb{R}^{d\times m} \rightarrow \{-1, 1\}^k$ that can

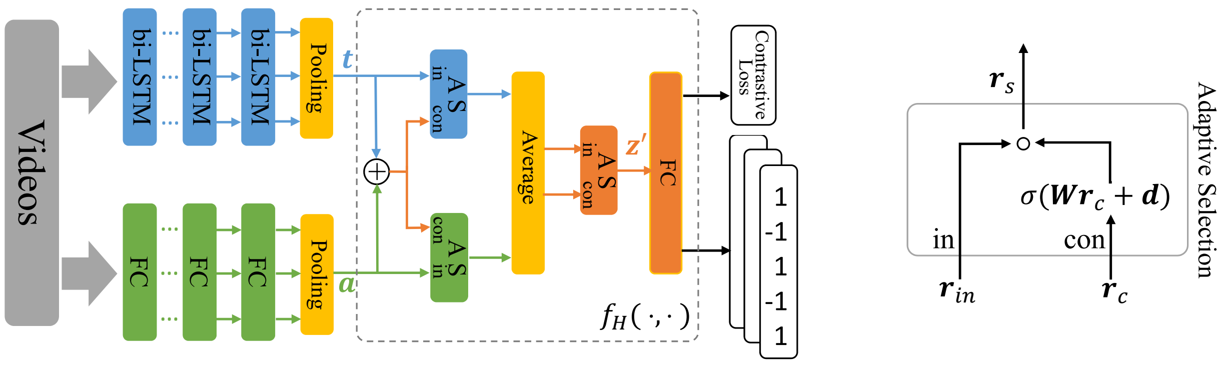

FIGURE 6.1: Illustration of the dual-stream framework (left) and the proposed Adaptive Selection mechanism (right). The top stream is composed of multiple stacked bi-directional LSTM layers, where the temporal patterns at video-level are captured. The bottom stream consists of multiple fully-connected layers, and the aim of this stream is to model the static visual features at frame-level. Followed by the proposed AS, the complementary advantages from the two deep representations, i.e. $\boldsymbol{a}$ and $\boldsymbol{t}$ are adaptively integrated to produce a finer representation $\boldsymbol{z}'$, based on which the final hash codes are generated.

encode video $\boldsymbol{V}$ into a k-bit binary code $\boldsymbol{b} \in \{-1, 1\}^k$, where $k \ll d$. In the meantime, the similarity information across each video pairs should be preserved in the compact hash codes. To this end, we employ the contrastive loss as in [28, 66]. It is naturally designed to pull the codes of similar videos together and push the codes of dissimilar videos away from each other. Specifically, the contrastive loss for video hashing is defined as follows:

$$\begin{aligned} L(\mathcal{V}, \mathcal{S}) = \sum_{i=1}^{n} \sum_{j=1}^{i-1} (s_{ij}\, D_H(\boldsymbol{b}_i, \boldsymbol{b}_j) \\ + (1 - s_{ij})\ \max(M - D_H(\boldsymbol{b}_i, \boldsymbol{b}_j), 0)) \end{aligned} \tag{6.1}$$

where M is the margin parameter and $D_H(\cdot, \cdot)$ denotes the Hamming distance between two binary vectors. The binary code $\boldsymbol{b}_i$ is generated by applying a sign function to the real-valued codes $\boldsymbol{z}_i \in (-1, 1)^k$ produced by the deep framework, i.e. $\boldsymbol{b}_i = sgn(\boldsymbol{z}_i)$, where:

$$sgn(\boldsymbol{z}) = \begin{cases} 1, & if\ \boldsymbol{z} \geq 0 \\ -1, & otherwise \end{cases} \tag{6.2}$$

However, it is infeasible to train a deep network with a sign function by using back-propagation, since the gradient of the sign function is zero in almost all cases. Hence, we replace the Hamming distance between binary codes in Eq. (6.1) with the Euclidean distance between real-valued codes. The relaxed loss function is written as:

$$\begin{aligned} L(\mathcal{V}, \mathcal{S}) = \sum_{i=1}^{n} \sum_{j=1}^{i-1} (s_{ij}\ D_E(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ + (1 - s_{ij})\ \max(M - D_E(\boldsymbol{z}_i, \boldsymbol{z}_j), 0)). \end{aligned} \tag{6.3}$$

Existing video hashing methods usually either solely consider static visual features [7, 92, 126] or only focus on modelling temporal patterns [123, 130]. Li *et al.* [59] proposed jointly modelling these two aspects; however, in their method the hash codes corresponding to each aspect are simply concatenated. How to effectively integrate these two aspects and maximize their utility is a problem yet to be solved. To solve the above issues, we propose a dual-stream deep hashing framework to adaptively model static visual features and temporal patterns. One stream is for static visual features, in which we construct a non-linear transformation function: $f_A : \mathbb{R}^{d \times m} \to (-1, 1)^{k'}$. By applying this function on a video $\boldsymbol{V}_i$, we aim to get a representation:

$$\boldsymbol{a}_i = f_A(\boldsymbol{V}_i) \tag{6.4}$$

which carries the necessary components with respect to static visual features for video hashing. The other stream is for temporal patterns. Similarly, we build another transformation function: $f_T : \mathbb{R}^{d \times m} \to (-1, 1)^{k'}$ to produce a representation:

$$\boldsymbol{t}_i = f_T(\boldsymbol{V}_i) \tag{6.5}$$

which contains the necessary components with respect to temporal patterns.

Finally, to effectively exploit the information from the above two aspects, we design a novel Adaptive Selection mechanism that enables the hash layer f_H to integrate these two deep representations complementarily, i.e. $\boldsymbol{a}_i$ and $\boldsymbol{t}_i$, to generate real-valued codes $\boldsymbol{z}_i \in (-1, 1)^k$:

$$\boldsymbol{z}_i = f_H(\boldsymbol{a}_i, \boldsymbol{t}_i). \tag{6.6}$$

In the following subsections, we will present the details of the proposed dual-stream network and Adaptive Selection mechanism. An overview of the proposed framework is illustrated in Figure 6.1.

6.2.2 Modelling Static Visual Features

To utilize the static visual features that exist in frames, we stack multiple fully-connected (FC) layers with a *tanh* activation to implement the transformation function f_A. Let $g_M(\cdot)$ denote the stacked FC layers that work on frames, and by feeding a frame $\boldsymbol{x}$ to $g_M(\cdot)$, we can get a deep representation for this frame, i.e.

$$\boldsymbol{e}^t = g_M(\boldsymbol{x}^t) \tag{6.7}$$

where $\boldsymbol{e}^t \in (-1,1)^{k'}$. In contrast to existing deep hashing methods for image retrieval, which take a single image as input, our transformation function f_A is expected to deal with a video containing a sequence of frames. Accordingly, f_A, which works on videos, is defined as:

$$\boldsymbol{a}_i = f_A(\boldsymbol{V}) = \frac{1}{m}\sum_{t=1}^{m} \boldsymbol{e}^t. \tag{6.8}$$

Note that, in Eq. (6.8), instead of applying $g_M(\cdot)$ on averaged frame features (i.e. $g_M(\frac{1}{m}\sum_{t=1}^{m}\boldsymbol{x}^t)$), we average the deep representation of each frame, i.e. $\boldsymbol{e}^t$ as the representation of a video. We adopt this strategy because we expect our framework can exploit the detail of every single frame and thus avoid information loss during feature pooling. This strategy is also employed in [59, 92].

6.2.3 Modelling Temporal Patterns

To model temporal patterns in videos, we employ bi-directional LSTM (bi-LSTM) [27] as the building block for constructing the transformation function f_T. By using bi-LSTM we aim to model the intact temporal context for each frame, while uni-directional LSTM can only capture the "previous" context by going forward or the "future" context by going backward.

Specifically, we stack multiple bi-LSTM layers to capture the temporal pattern in a video. We average the outputs of all timesteps of the final bi-LSTM layer as the final deep representation with respect to a temporal pattern:

$$\boldsymbol{t}_i = f_T(\boldsymbol{V}) = \frac{1}{m}\sum_{t=1}^{m} \bar{\boldsymbol{h}}^t \tag{6.9}$$

where $\bar{\boldsymbol{h}}^t = \frac{1}{2}(\boldsymbol{h}^t + \hat{\boldsymbol{h}}^t)$, $\bar{\boldsymbol{h}}^t \in (-1,1)^{k'}$ is the output of the last bi-LSTM layer at timestep t, $\boldsymbol{h}^t$ and $\hat{\boldsymbol{h}}^t$ denote the outputs at timestep t of the forward LSTM and the backward LSTM respectively.

The detailed implementation of the basic LSTM cell for the forward direction is as below the (backward direction uses the same cell):

$$\boldsymbol{i}^t = \sigma(\boldsymbol{W}_i \boldsymbol{x}^t + \boldsymbol{U}_i \boldsymbol{h}^{t-1} + \boldsymbol{d}_i) \tag{6.10a}$$

$$\boldsymbol{f}^t = \sigma(\boldsymbol{W}_f \boldsymbol{x}^t + \boldsymbol{U}_f \boldsymbol{h}^{t-1} + \boldsymbol{d}_f) \tag{6.10b}$$

$$\boldsymbol{o}^t = \sigma(\boldsymbol{W}_o \boldsymbol{x}^t + \boldsymbol{U}_o \boldsymbol{h}^{t-1} + \boldsymbol{d}_o) \tag{6.10c}$$

$$\boldsymbol{y}^t = \phi(\boldsymbol{W}_z \boldsymbol{x}^t + \boldsymbol{U}_z \boldsymbol{h}^{t-1} + \boldsymbol{d}_z) \tag{6.10d}$$

$$\boldsymbol{c}^t = \boldsymbol{y}^t \circ \boldsymbol{i}^t + \boldsymbol{c}^{t-1} \circ \boldsymbol{f}^t \tag{6.10e}$$

$$\boldsymbol{h}^t = \phi(\boldsymbol{c}^t) \circ \boldsymbol{o}^t \tag{6.10f}$$

where $\boldsymbol{W}$ are the input weight matrices, $\boldsymbol{U}$ are the recurrent weight matrices, and $\boldsymbol{d}$ are the bias vectors. $\boldsymbol{i}^t$, $\boldsymbol{f}^t$ and $\boldsymbol{o}^t$ denote the input gate, forget gate and output gate respectively. $\sigma(\cdot)$ is the *sigmoid* recurrent gate activation function, $\phi(\cdot)$ is the *tanh* input activation function and both $\sigma(\cdot)$ and $\phi(\cdot)$ are element-wise functions.

6.2.4 Adaptive Selection

For video hashing, it is important to learn high-quality representations of videos before the final hashing step. It has also been proved that different components in deep representations have their own semantic or conceptual meanings [20, 112, 129] and they respond differently in different visual tasks, e.g. classification and scene recognition. Motivated by the above facts, after having the deep representations $\boldsymbol{a}_i$ and $\boldsymbol{t}_i$ (corresponding to static visual feature and temporal pattern, respectively) in hand, we aim to adaptively integrate the complementary components from both representations to generate better hash codes.

To this end, we propose Adaptive Selection, which is implemented with a gating mechanism [16, 31]. For a given input representation $\boldsymbol{r}_{in} \in \mathbb{R}^{d1}$, we want to pick up the useful component by consulting the auxiliary conditional representation $\boldsymbol{r}_c \in \mathbb{R}^{d2}$. The representation of the selected component, i.e. $\boldsymbol{r}_s \in \mathbb{R}^{d1}$ is generated by the following adaptive selective function $f_S(\cdot, \cdot)$:

$$\begin{aligned} \boldsymbol{r}_s &= f_S(\boldsymbol{r}_{in}, \boldsymbol{r}_c) \\ &= \sigma(\boldsymbol{W}\boldsymbol{r}_c + \boldsymbol{d}) \circ \boldsymbol{r}_{in} \end{aligned} \tag{6.11}$$

where $\sigma(\cdot)$ is a *sigmoid* activation function, and $\boldsymbol{W} \in \mathbb{R}^{d1\times d2}$ and $\boldsymbol{d} \in \mathbb{R}^{d1}$ are the parameters to be trained. The gating vector $\sigma(\boldsymbol{W}\boldsymbol{r}_c + \boldsymbol{d}) \in (0,1)^{d1}$ controls the ratio in which the information carried by each element of $\boldsymbol{r}_{in}$ should be passed out, by consulting the conditional representation $\boldsymbol{r}_c$. A demonstration of AS is provided in Figure 6.1. Although the deep representations, i.e. $\boldsymbol{a}_i$ and $\boldsymbol{t}_i$, are generated by two independent streams that are dedicated to temporal patterns and static visual features respectively, it is inevitable that $\boldsymbol{a}_i$ and $\boldsymbol{t}_i$ contain redundant information as they are both derived from the same original video frame features. We expect the selected components to be complementary to each other so that the integrated deep representation used by the hash function would be compact and solid. To this end, we allow the AS to investigate both of $\boldsymbol{a}_i$ and $\boldsymbol{t}_i$, then it can accordingly decide what components are necessary and should be selected for each aspect:

$$\begin{aligned} \hat{\boldsymbol{a}}_i &= f_S(\boldsymbol{a}_i, \boldsymbol{a}_i \oplus \boldsymbol{t}_i) \\ \hat{\boldsymbol{t}}_i &= f_S(\boldsymbol{t}_i, \boldsymbol{a}_i \oplus \boldsymbol{t}_i) \end{aligned} \tag{6.12}$$

in which, $\hat{\boldsymbol{a}}_i$ and $\hat{\boldsymbol{t}}_i \in (-1,1)^{k'}$ are the selected components for static visual features and temporal patterns respectively, and $\oplus$ denotes a concatenation operation. The hash layer, i.e. Eq. (6.6) is defined as the combination of Eq. (6.12), Eq. (6.12) and the following equations:

$$\begin{aligned} \hat{\boldsymbol{at}}_i &= \frac{1}{2}(\hat{\boldsymbol{a}}_i + \hat{\boldsymbol{t}}_i) \\ \boldsymbol{z}'_i &= f_S(\hat{\boldsymbol{at}}_i, \hat{\boldsymbol{at}}_i,) \\ \boldsymbol{z}_i &= g(\boldsymbol{z}'_i) \end{aligned} \tag{6.13}$$

where $\boldsymbol{z}'_i \in (-1,1)^{k'}$, $\boldsymbol{z}_i \in (-1,1)^k$ and $g(\cdot)$ denotes a fully-connected layer with *tanh* activation. Here we apply AS to the intermediate representation $\hat{\boldsymbol{at}}_i$ to allow it to do a self-investigation for adaptively integrating the selected components from each aspect, which is supervised by the overall hashing objective function as in Eq. (6.3). The integrated representation $\boldsymbol{z}'_i$ is fed to $g(\cdot)$ to generate real-valued hash codes $\boldsymbol{z}_i$. Note that, in Eq. (6.13), instead of concatenation, we use element-wise average to combine $\hat{\boldsymbol{a}}_i$ and $\hat{\boldsymbol{t}}_i$. Here we aim to capture the element-wise (or bit-wise) correspondence between the selected components, i.e. $\hat{\boldsymbol{a}}_i$ and $\hat{\boldsymbol{t}}_i$, from the two aspects, since the representation $\boldsymbol{z}'_i$ integrated from them is directly used by the hash function $g(\cdot)$.

Now we have all the building blocks to construct our framework, and the overall flow path is demonstrated in Algorithm 2.

Algorithm 2 The encoding process for a given video.

```
Input: Video $V_i$, and the learned model: $f_A$, $f_T$, $g$, $f_S^{1-3}$;
Output: hash code $\boldsymbol{b}_i$;
    Generate deep representations for two aspects:
    1: $\boldsymbol{a}_i \leftarrow f_A(\boldsymbol{V}_i^a)$;
    2: $\boldsymbol{t}_i \leftarrow f_T(\boldsymbol{V}_i^t)$;
    Apply AS to deep representations:
    3: $\hat{\boldsymbol{a}}_i = f_S^1(\boldsymbol{a}_i, \boldsymbol{a}_i \oplus \boldsymbol{t}_i)$;
    4: $\hat{\boldsymbol{t}}_i = f_S^2(\boldsymbol{t}_i, \boldsymbol{a}_i \oplus \boldsymbol{t}_i))$;
    5: $\hat{\boldsymbol{at}}_i = \frac{1}{2}(\hat{\boldsymbol{a}}_i + \hat{\boldsymbol{t}}_i)$;
    6: $\boldsymbol{z}_i' = f_S^3(\hat{\boldsymbol{at}}_i, \hat{\boldsymbol{at}}_i,)$;
    Generate real-valued hash code:
    7: $\boldsymbol{z}_i = g(\boldsymbol{z}_i')$;
    Binarization:
    8: $\boldsymbol{b}_i = sgn(\boldsymbol{z}_i)$;
    9: return $\boldsymbol{b}_i$;
```

6.3 Experiments

6.3.1 Experiment Settings

Datasets. We evaluate our framework on two large-scale video datasets, i.e. FCVID and ActivityNet. Their brief summaries are provided as follows. For more details we refer readers to [30, 43]. FCVID [43] consists of 91,223 Web videos annotated manually into 239 categories covering a wide range of topics like social events (e.g. “tailgate party”), procedural events (e.g. “making cake”), objects (e.g. “panda”), scenes (e.g. “beach”), etc. The average duration per video is 167 seconds. We use its standard split of 45,611 videos for training and 45,612 videos for retrieval test. ActivityNet [30] is a recently released benchmark video dataset, which contains a wide range of complex human activities that are of interest to people in their daily living. It comprises 28K of videos of 203 activity categories. The lengths of the videos range from several minutes to half an hour. Many of the videos are shot by amateurs in uncontrolled environments, where the variances within the same activity category are often large. ActivityNet provides trimmed and untrimmed videos for evaluation. We adopt the more challenging untrimmed videos to evaluate our framework.

Evaluation Protocols. We employ Average Precision at top K retrieved videos (AP@K) for retrieval performance evaluation as in [59, 130]. For a given query, the retrieved videos are ranked by their Hamming distances to the query. The definition of AP@K is as: $\frac{1}{min(R,K)} \sum_{i=1}^{K} \frac{R_i}{i} \times I_i$, where R_i is the number of relevant videos in the top i videos, R is the number of relevant videos in the database and $I_i = 1$ if the i-th video is relevant and 0 otherwise. Two videos are considered as relevant if they belong to the same class. For fair comparison, we adopt the same protocol as in [59]: From each dataset, 10 videos per category are randomly sampled as queries. As a result, we get around 2000 queries for each dataset. Then the mean of AP@Ks (mAP@K) of all the queries is used as the overall metric for the corresponding dataset. For the FCVID dataset, we sample queries from the test set, and the rest of the test set is employed as the database. For the ActivityNet dataset, the labels of test set are not publicly available, as the authors are reserving the test data for a potential future competition. Hence, we sample queries from the validation set, then the rest of the validation set and the training set is used as the database.

Implementation Details. To generate a k-bit hash code, we use stacked bi-LSTM layers for the stream that models temporal patterns. In each layer, the LSTM cells in each direction have k' units. The k'-dimensional outputs from two directions are averaged as the output of the bi-LSTM. In the stream modelling static visual features, we use multiple FC layers. The first layer has $\frac{d}{2}$ units, and the following layers have k' units. The two k'-dimensional deep representations produced by these two streams are then fed to the hash function f_H, which outputs k-dimensional real-values hash codes. In the following experiments, without specification, we use three stacked FC layers and three stacked bi-LSTM layers and set $k' = 2k$ as a compromise between performance and computing cost. The margin parameter M is empirically chosen from $[3, 5, 7, 10, 12]$ by cross-validation. As a compromise for GPU memory and training time, we uniformly sample 30 frames for each video, as in [59]. For each frame, we employ ResNet [29] to extract a 2048-D CNN feature as its representation. Note that the ResNet feature is employed for all compared methods to achieve a fair comparison. The stochastic gradient descent algorithm with momentum is used to train our network and the learning rate, momentum, and dropout rate (applied to LSTM layers) are set to 0.01, 0.9 and 0.1, respectively.

We use batch size 150 and online pair generation as described in [66]. Training continues until validation loss ceases to decrease or after 300 epochs.

6.3.2 Empirical Analysis

Method	ActivityNet				FCVID			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
SS-A	0.1770	0.2329	0.2767	0.3452	0.1962	0.2720	0.3604	0.3737
SS-T	0.1696	0.2127	0.3019	0.3503	0.2044	0.2746	0.3560	0.3886
DS-NoAS	0.2378	0.2890	0.3534	0.4081	0.2624	0.2995	0.3881	0.4222
DS-AS	**0.2715**	**0.3392**	**0.4020**	**0.4600**	**0.2924**	**0.3424**	**0.4358**	**0.4740**

Table 6.1: Comparison with baseline methods on the FCVID and ActivityNet datasets with mAP@20 as metric. SS-A solely utilizes the static visual feature. SS-T only models the temporal information. DS-NoAS concatenates the two aspects. DS-AS adaptively integrates the two aspects to exploit the complementary advantages of them for video event retrieval.

A comprehensive empirical analysis is conducted to verify the effectiveness of the proposed framework. In this section, we examine to what extent the proposed Adaptive Selection mechanism can promote the performance of video hashing. To this end, we compare our full model, i.e. the dual-stream network equipped with Adaptive Selection, denoted as **DS-AS**, with the following baseline methods: (1) **SS-A**, single stream network, which only captures the static visual features (as described in Section 6.2.2). (2) **SS-T**, single stream network, which only models the temporal patterns (as illustrated in Section 6.2.3). (3) **DS-NoAS**, dual-stream network without AS.

Firstly, we show the hashing performance of the above methods on the two datasets (i.e. ActivityNet and FCVID) in Table 6.1, where mAP@20 is employed as the evaluation metric. In practical retrieval systems, users commonly pay more attention to the higher ranking results, i.e. those at the top of the returned ranking list, hence mAP@20 is an important metric for verifying hashing methods. The results regarding hash code lengths [16, 32, 64, 128] are reported. As the results show, DS-NoAS achieves better performance than both SS-A and SS-T which indicates that combining

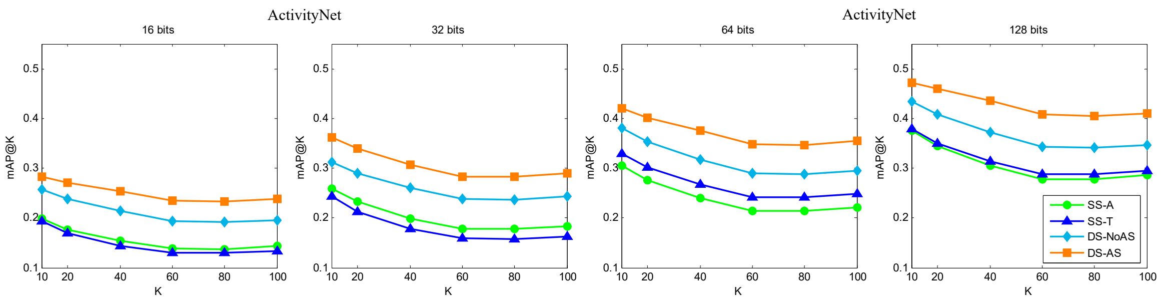

FIGURE 6.2: Comparison with baseline methods on the ActivityNet dataset.

static visual features and temporal patterns leads to a better hashing performance. This observation is also consistent with the studies in [59]. More importantly, our full model DS-AS outperforms all other baseline methods by clear margins, for all hash code lengths. This not only confirms our hypothesis that static visual features and temporal patterns do not always contribute equally to hashing performance, instead, they should be integrated adaptively, but it also proves the effectiveness of the proposed Adaptive Selection mechanism. We analyze the reason why the AS works better than simply concatenation, i.e. DS-NoAS, as follows. Firstly, the AS integrates our two streams, which enables them to interact with each other to exploit the complementary advantages of static visual feature and temporal pattern. The two streams in DS-NoAS are totally separated without interaction. Secondly, by exploiting the complementary advantages of the two above aspects, the AS outputs an compact video representation and discards redundant information. Therefore, The AS can outperform DS-NoAS.

Secondly, we visualize the performance metric mAP@K against $K \in [10, 20, 40, 60, 80, 100]$ for a more comprehensive analysis. The results for the ActivityNet and FCVID datasets are shown in Figure 6.2 and Figure 6.3, respectively. As the figures show, on both datasets, DS-AS outperforms DS-NoAS with remarkable gaps at all code lengths in [16, 32, 64, 128] which again confirms the effectiveness of the proposed AS mechanism. We also observed that DS-AS achieves greater performance over DS-NoAS for larger hash code lengths. We analyze the reason as follows. The farthest distance in the Hamming space is limited by the number of bits and therefore, the representative ability of the integrated representation produced by AS is also constrained by the number of bits. Hence, DS-AS can performs better with larger hash code lengths.

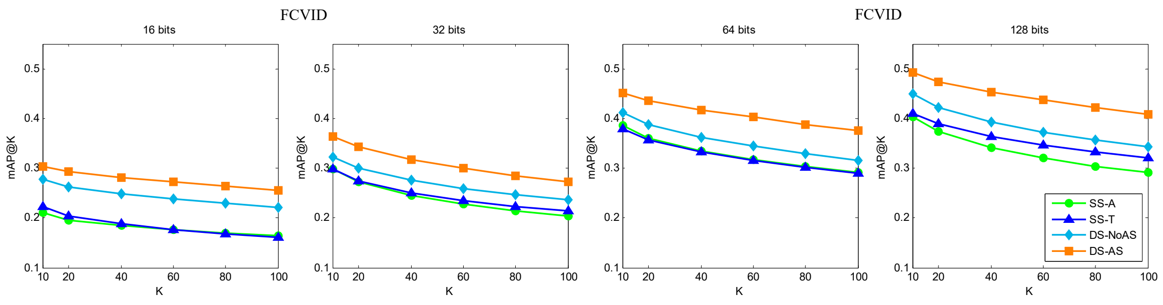

Figure 6.3: Comparison with baseline methods on the FCVID dataset.

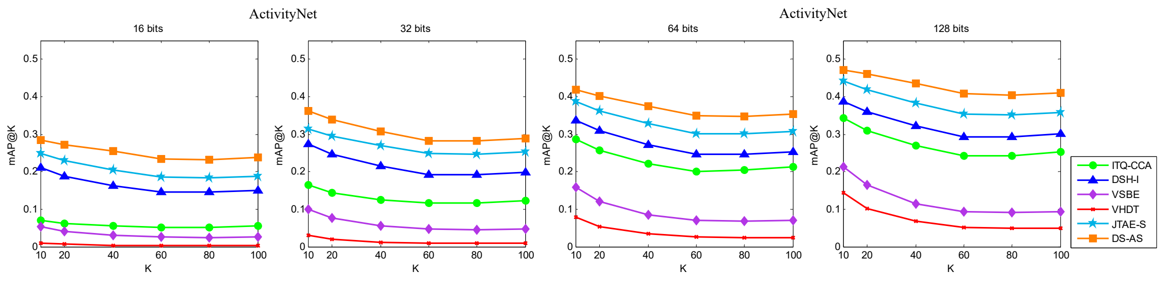

Figure 6.4: Comparison with state-of-the-art methods on the ActivityNet dataset.

6.3.3 Comparison with State-of-the-art Methods

In this section, we compare our framework with two types of state-of-the-art methods. We use “*” to denote supervised methods.

Image Hashing with Video Frames. The first type is image hashing methods, i.e. SH [110], ITQ [25], ITQ-CCA* [25], and DSH-I* [66]. We apply these methods to video frames, thus the temporal information is neglected. By comparing our model with these methods, we expect to discover to what extent the temporal patterns contained in videos can boost the hashing performance. SH [110] and ITQ [25] are classical unsupervised hashing methods and ITQ-CCA* [25] is the supervised version of ITQ. DSH-I* [66] is a recently proposed supervised deep hashing methods that utilizes semantic label information for image retrieval.

Video Hashing. The second type of methods include hashing methods that are dedicated to videos, i.e. MFH [92] VSBE* [126], VHDT* [123], BLSTM [130], JTAE [59], and JTAE-S* [59]. We aim to confirm the effectiveness of our framework and the proposed Adaptive Selection mechanism through this comparison. MFH [92] is designed for preserving the pair-wise similarities between

Method	ActivityNet				FCVID			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
SH [110]	0.0262	0.0640	0.1001	0.1298	0.0439	0.1041	0.1605	0.2084
ITQ [25]	0.0259	0.0491	0.0909	0.1220	0.0546	0.1220	0.1637	0.2066
ITQ-CCA* [25]	0.0611	0.1438	0.2568	0.3097	0.0772	0.1521	0.2496	0.3290
DSH-I* [66]	0.1875	0.2473	0.3086	0.3591	0.2098	0.2821	0.3725	0.3818
MFH [92]	0.0255	0.0522	0.0920	0.1320	0.0616	0.1180	0.1746	0.2264
VSBE* [126]	0.0405	0.0765	0.1204	0.1639	0.0661	0.1362	0.2120	0.2834
VHDT* [123]	0.0064	0.0193	0.0536	0.1026	0.0126	0.0459	0.1255	0.2241
BLSTM [130]	0.0206	0.0528	0.0914	0.1337	0.0598	0.1184	0.1839	0.2380
JTAE [59]	0.0256	0.0554	0.1044	0.1502	0.0614	0.1330	0.2008	0.2631
JTAE-S* [59]	0.2290	0.2955	0.3629	0.4179	0.2363	0.2852	0.3935	0.4392
DS-AS*	**0.2715**	**0.3392**	**0.4020**	**0.4600**	**0.3317**	**0.3424**	**0.4358**	**0.4740**

TABLE 6.2: Comparison with state-of-the-art methods on ActivityNet and FCVID datasets, with mAP@20 as metric. * denotes supervised methods. The top four rows are image hashing methods applied to video frames, and the bottom seven rows are video hashing methods.

frames in the original feature space, whereas, VSBE* [126] is a semi-supervised methods based on selecting representative frames and VHDT* [123] exploits pair-wise frame order for video hashing. BLSTM [130] extracts temporal patterns in videos by Binary LSTM cells. JTAE [59] jointly models temporal patterns and static visual features by imposing constraints with respect to hashing criteria on the concatenation of the two aspects, and JTAE-S* [59] is implemented by replacing the original reconstruction loss with contrastive loss as in Eq. (6.3), where the constraints are kept.

The comparison of our model with all the above state-of-the-art methods is reported in Table 6.2, where mAP@20 is used as the overall evaluation metric. For further comparison we also plot the performance of all supervised methods in Figure 6.4 and 6.5, with mAP@K against different K values. As the results show, our method, i.e. DS-AS*, outperforms all the other methods with clear margins on both the ActivityNet and FCVID datasets. For the image hashing methods, DSH-I* achieves the best performance. For the video hashing methods, JTAE-S* outperforms DSH-I* by considering both

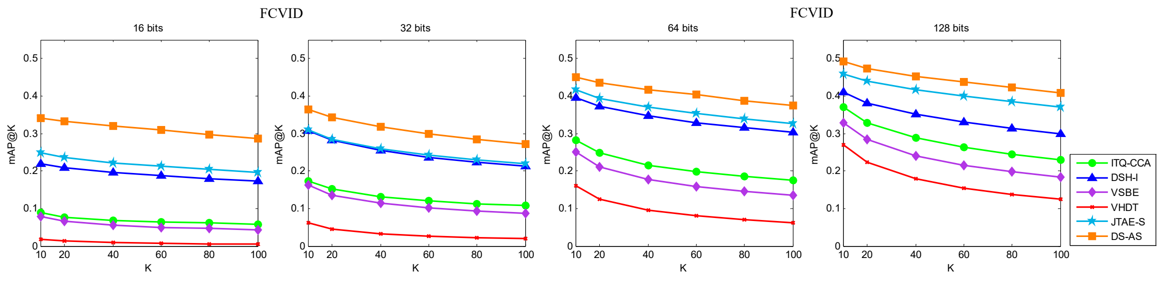

Figure 6.5: Comparison with state-of-the-art methods on the FCVID dataset.

temporal patterns and static visual features, but our method outperforms JTAE-S* by exploiting the complementary advantages of these two aspects. This again proves the effectiveness of the proposed Adaptive Selection mechanism.

6.4 Summary

In this chapter, to tackle the challenging video hashing task, we proposed to effectively exploit the complementary advantages from two essential aspects of videos, i.e. temporal patterns and static visual features (corresponding to semantic information). To this end, we developed a dual-stream deep framework to model the two aspects by each stream, respectively. We designed a novel Adaptive Selection mechanism, which enables our framework to adaptively integrate the complementary components from each aspect and consequently generate a finer deep binary representation for videos.

Recall the discussion in Section 5.5, the semantic information and temporal information are collectively utilized by two concatenated encoders and manually imposed learning criteria. To further adaptively exploit the two types of information, in this chapter, the proposed AS mechanism enables them to interact and cooperate with each other, therefore, the complementary advantages of semantic information and temporal pattern can be utilized more effectively.

Extensive experiments conducted on two large-scale benchmark video datasets validated the effectiveness of the proposed AS mechanism and the superiority of our hashing framework over the state-of-the-art methods.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we exploit semantic and temporal information for video event understanding. We focus on video event classification and retrieval, which are two main tasks related to video event understanding.

For video event classification, in chapter 3, we proposed to discover the hierarchical latent concepts for video event classification utilizing underlying semantic cues. Two levels of latent concepts were introduced, frame-level static-visual concepts and segment-level activity concepts. A hierarchical structure was proposed to model these latent concepts. Our model abstracts the underlying semantic information into latent concepts. A max-margin framework is employed for model learning, then we develop an alternative linear programming algorithm for latent concepts inference. In contrast to two-stage frameworks, the proposed model requires no effort to construct and maintain a concept database, and the latent concepts are discovered adaptively based on the underlying semantic cues. Thus, our model does not encounter the error propagation problem that occurred in two-stage frameworks. Furthermore, the concepts discovered by our model are not only helpful for explaining the event classification results but also can be used to build a semantic-aware index for event videos. In chapter 4, we proposed a framework with a novel attention model to automatically utilize weak semantic relevance to assist in the video classification task. This framework jointly optimizes two objectives at video-level and shot-level separately, which explicitly affect video classification from both global-level (i.e. video-level labels) and local-level (i.e. shot-level attention scores). To alleviate the effect of the noise carried by weak semantic relevance, we use weak semantic relevance as a weak guidance in the proposed attention model, instead of considering it as the attention score directly. This process significantly improves the effectiveness of our proposed model.

For video event retrieval, in chapter 5, we proposed a novel unsupervised video hashing model. In contrast to existing conventional hash methods that only utilize static visual appearance or solely focus on temporal patterns, the proposed model exploits both of these types of information by collectively learning two encoders, i.e. the temporal encoder and the appearance encoder. Our model aims to maximize the utility of temporal patterns and static visual appearance by imposing three learning criteria on the two encoders directly and strictly. In this way, redundant information captured by the two encoders can be reduced and as a results, high quality hash codes can be generated. Essentially, our model is an end-to-end framework which can be easily extended for multi-view or multi-modal hashing. Moving on from this, query and search between data in different formats is a potential future research direction. For example, we can search videos by photos shot by mobile devices. In chapter 6, to tackle the challenging video hashing task, we proposed to effectively exploit the complementary advantages from two essential aspects of videos, i.e. temporal patterns and static visual features. To this end, we developed a dual-stream deep framework to model the two aspects by each stream, respectively. Finally, we designed a novel Adaptive Selection mechanism, which enables our framework to adaptively integrate the complementary components from each aspect and consequently generate a finer deep binary representation for videos.

7.2 Future Work

In thesis, pre-trained CNNs are used to extract a feature representation for each frame in videos. This means the CNN has taken over the modeling of the spatial pattern in a frame. However, the spatial patterns of entities, e.g. objects, people, animals and scene, are important for video event understanding. And the pre-trained CNNs can not comprehensively exploit the spatial patterns in videos. How to effectively integrate semantic, temporal and spatial information to facilitate video event understanding is a challenging task for future research.

Hashing methods are wildly used for retrieval tasks because of the low storage cost and high retrieval efficiency of hash codes. And in this thesis, we have discussed hashing methods for video event retrieval. Besides, hashing techniques can also be applied on classification tasks. How to incorporate hashing techniques for video event classification is also an promising research topic.

References

[1] R. Arandjelovic and A. Zisserman. All about VLAD. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013.

[2] A. Babenko and V. S. Lempitsky. Aggregating deep convolutional features for image retrieval. In *IEEE International Conference on Computer Vision*, 2015.

[3] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[4] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, and M. Shah. Recognition of complex events: Exploiting temporal dynamics between underlying concepts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2243–2250, 2014.

[5] S. Boyd and L. Vandenberghe. Convex optimization. *Cambridge University Press*, 2004.

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA*, pages 327–336, 1998.

[7] L. Cao, Z. Li, Y. Mu, and S. Chang. Submodular video hashing: a unified framework towards video pooling and indexing. In *ACM Conference on Multimedia*, pages 299–308, 2012.

[8] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *IEEE International Conference on Computer Vision*, 2017.

[9] X. Chang, Y. Yang, G. Long, C. Zhang, and A. G. Hauptmann. Dynamic concept composition for zero-example event detection. In *AAAI*, pages 3464–3470, 2016.

[10] X. Chang, Y. Yang, E. P. Xing, and Y. Yu. Complex event detection using semantic saliency and nearly-isotonicSVM. In *International Conference on Machine Learning*, 2015.

[11] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference, BMVC 2014, Nottingham, UK*, 2014.

[12] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[14] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *9th European Conference on Computer Vision*, pages 428–441, 2006.

[15] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):5:1–5:60, 2008.

[16] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016.

[17] T. Do, A. Doan, and N. Cheung. Learning to hash with binary deep neural network. In *14th European Conference on Computer Vision*, pages 219–234, 2016.

[18] T. M. T. Do and T. Artières. Large margin training for hidden Markov models with partially observed states. In *International Conference on Machine Learning*, pages 265–272, 2009.

[19] T. M. T. Do and T. Artières. Regularized bundle methods for convex and non-convex risks. *Journal of Machine Learning Research*, 13:3539–3583, 2012.

[20] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pages 647–655, 2014.

[21] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010.

[22] C. Gan, N. Wang, Y. Yang, D. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2568–2577, 2015.

[23] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[24] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[25] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 817–824, 2011.

[26] A. Graves, N. Jaitly, and A. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013.

[27] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610, 2005.

[28] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1735–1742, 2006.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[30] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[32] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen. Hashing with angular reconstructive embeddings. *IEEE Transactions on Image Processing*, 2017.

[33] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[34] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. In *12th European Conference on Computer Vision*, pages 430–444, 2012.

[35] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2555–2562, 2013.

[36] M. Jain, J. C. van Gemert, T. Mensink, and C. G. M. Snoek. Objects2action: Classifying and localizing actions without any video example. In *IEEE International Conference on Computer Vision*, pages 4588–4596, 2015.

[37] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[38] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Conference on Multimedia*, pages 675–678, 2014.

[39] L. Jiang, A. G. Hauptmann, and G. Xiang. Leveraging high-level and low-level features for multimedia event detection. In *ACM Conference on Multimedia*, pages 449–458, 2012.

[40] L. Jiang, S. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann. Fast and accurate content-based semantic search in 100m internet videos. In *ACM Conference on Multimedia*, pages 49–58, 2015.

[41] Y. Jiang, S. Bhattacharya, S. Chang, and M. Shah. High-level event recognition in unconstrained videos. *International Journal of Multimedia Information Retrieval*, 2(2):73–101, 2013.

[42] Y. Jiang, Q. Dai, X. Xue, W. Liu, and C. Ngo. Trajectory-based modeling of human actions with motion reference points. In *12th European Conference on Computer Vision*, pages 425–438, 2012.

[43] Y. Jiang, Z. Wu, J. Wang, X. Xue, and S. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2):352–364, 2018.

[44] Y. Jiang, G. Ye, S. Chang, D. P. W. Ellis, and A. C. Loui. Consumer video understanding: a benchmark database and an evaluation of human and machine performance. In *ACM International Conference on Multimedia Retrieval*, page 29, 2011.

[45] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[46] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[48] N. Krüger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. H. Piater, A. J. Rodríguez-Sánchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1847–1871, 2013.

[49] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.

[50] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.

[51] K. Lai, D. Liu, M. Chen, and S. Chang. Recognizing complex events in videos by learning key static-dynamic evidences. In *13th European Conference on Computer Vision*, pages 675–688, 2014.

[52] K. Lai, F. X. Yu, M. Chen, and S. Chang. Video event detection by inferring temporal instance labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2251–2258, 2014.

[53] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2–3):107–123, 2005.

[54] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3361–3368, 2011.

[55] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[56] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *TOMCCAP*, 2(1):1–19, 2006.

[57] C. Li, J. Cao, Z. Huang, L. Zhu, and H. T. Shen. Leveraging weak semantic relevance for complex video event classification. In *IEEE International Conference on Computer Vision*, 2017.

[58] C. Li, Z. Huang, Y. Yang, J. Cao, X. Sun, and H. T. Shen. Hierarchical latent concept discovery for video event detection. *IEEE Trans. Image Processing*, 26(5):2149–2162, 2017.

[59] C. Li, Y. Yang, J. Cao, and Z. Huang. Jointly modeling static visual appearance and temporal pattern for unsupervised video hashing. In *ACM International Conference on Information and Knowledge Management*, 2017.

[60] W. Li, S. Wang, and W. Kang. Feature learning based deep supervised hashing with pairwise labels. In *International Joint Conference on Artificial Intelligence*, pages 1711–1717, 2016.

[61] W. Li, Q. Yu, A. Divakaran, and N. Vasconcelos. Dynamic pooling for complex event recognition. In *IEEE International Conference on Computer Vision*, pages 2728–2735, 2013.

[62] W. Li, Q. Yu, H. S. Sawhney, and N. Vasconcelos. Recognizing activities via bag of words for attribute dynamics. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2587–2594, 2013.

[63] J. Liang, L. Jiang, D. Meng, and A. G. Hauptmann. Learning to detect concepts from webly-labeled video data. In *International Joint Conference on Artificial Intelligence*, pages 1746–1752, 2016.

[64] R. Lin, D. A. Ross, and J. Yagnik. SPEC hashing: Similarity preserving algorithm for entropy-based coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 848–854, 2010.

[65] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

[66] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2064–2072, 2016.

[67] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2074–2081, 2012.

[68] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *International Conference on Machine Learning*, pages 1–8, 2011.

[69] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[70] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, and H. T. Shen. Robust discrete code modeling for supervised hashing. *Pattern Recognition*, 2017.

[71] S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff. Do less and achieve more: Training cnns for action recognition utilizing action images from the web. *Pattern Recognition*, 2017.

[72] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[73] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *CoRR*, abs/1706.06905, 2017.

[74] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[75] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[76] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[77] P. Natarajan, S. Wu, S. N. P. Vitaladevuni, X. Zhuang, S. Tsakalidis, U. Park, R. Prasad, and P. Natarajan. Multimodal feature fusion for robust event detection in web videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1298–1305, 2012.

[78] J. C. Niebles, C. Chen, and F. Li. Modeling temporal structure of decomposable motion segments for activity classification. In *11th European Conference on Computer Vision*, pages 392–405, 2010.

[79] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *International Conference on Machine Learning*, pages 353–360, 2011.

[80] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[81] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[82] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[83] V. Ramanathan, K. D. Tang, G. Mori, and L. Fei-Fei. Learning temporal embeddings for complex video analysis. In *IEEE International Conference on Computer Vision*, 2015.

[84] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[85] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1234–1241, 2012.

[86] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681, 1997.

[87] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.

[88] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[89] B. Singh, X. Han, Z. Wu, V. I. Morariu, and L. S. Davis. Selecting relevant web trained concepts for automated event retrieval. In *IEEE International Conference on Computer Vision*, pages 4561–4569, 2015.

[90] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision*, pages 1470–1477, 2003.

[91] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. C. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[92] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Conference on Multimedia*, pages 423–432, 2011.

[93] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015.

[94] C. Sun and R. Nevatia. ACTIVE: activity concept transitions in video event classification. In *IEEE International Conference on Computer Vision*, pages 913–920, 2013.

[95] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[96] A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. S. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3681–3688, 2012.

[97] K. D. Tang, F. Li, and D. Koller. Learning latent temporal structure for complex event detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1250–1257, 2012.

[98] B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006.

[99] L. Torresani, M. Szummer, and A. W. Fitzgibbon. Efficient object category recognition using classemes. In *11th European Conference on Computer Vision*, pages 776–789, 2010.

[100] TRECVID-MED. https://www.nist.gov/itl/iad/mig/med-2014-evaluation. 2014.

[101] TRECVID-MED11. http://www.nist.gov/itl/iad/mig/med11.cfm.

[102] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

[103] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008.

[104] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[105] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176, 2011.

[106] J. Wang, S. Kumar, and S. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

[107] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[108] L. Wang, Y. Qiao, and X. Tang. Latent hierarchical model of temporal structure for complex activity classification. *IEEE Transactions on Image Processing*, 23(2):810–822, 2014.

[109] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *14th European Conference on Computer Vision*, pages 20–36, 2016.

[110] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[111] WikiDump. https://dumps.wikimedia.org/enwiki/latest/.

[112] Z. Wu, Y. Fu, Y.-G. Jiang, and L. Sigal. Harnessing object and scene semantics for large-scale video understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[113] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, pages 2156–2162, 2014.

[114] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[115] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1798–1807, 2015.

[116] T. Yan, X. Xu, S. Guo, Z. Huang, and X. Wang. Supervised robust discrete multimodal hashing for cross-media retrieval. In *ACM International Conference on Information and Knowledge Management*, pages 1271–1280, 2016.

[117] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, and H. T. Shen. Zero-shot hashing via transferring supervised knowledge. In *ACM Conference on Multimedia*, pages 1286–1295, 2016.

[118] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Transactions on Big Data*, 1(4):162–171, 2015.

[119] Y. Yang, Y. Yang, Z. Huang, H. T. Shen, and F. Nie. Tag localization with spatial correlations and joint group sparsity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 881–888, 2011.

[120] Y. Yang, Z. Zha, Y. Gao, X. Zhu, and T. Chua. Exploiting web images for semantic video indexing via robust sample-specific loss. *IEEE Trans. Multimedia*, 16(6):1677–1689, 2014.

[121] L. Yao, A. Torabi, K. Cho, N. Ballas, C. J. Pal, H. Larochelle, and A. C. Courville. Describing videos by exploiting temporal structure. In *IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[122] G. Ye, Y. Li, H. Xu, D. Liu, and S. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM Conference on Multimedia*, pages 471–480, 2015.

[123] G. Ye, D. Liu, J. Wang, and S. Chang. Large-scale video hashing via structure learning. In *IEEE International Conference on Computer Vision*, pages 2272–2279, 2013.

[124] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659, 2016.

[125] C. J. Yu and T. Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning*, pages 1169–1176, 2009.

[126] L. Yu, Z. Huang, J. Cao, and H. T. Shen. Scalable video event retrieval by visual state binary embedding. *IEEE Trans. Multimedia*, 18(8):1590–1603, 2016.

[127] L. Yu, J. Shao, X. Xu, and H. T. Shen. Max-margin adaptive model for complex video pattern recognition. *Multimedia Tools Appl.*, 74(2):505–521, 2015.

[128] L. Yu, Y. Yang, Z. Huang, P. Wang, J. Song, and H. T. Shen. Web video event recognition by semantic analysis from ubiquitous documents. *IEEE Transactions on Image Processing*, 25(12):5689–5701, 2016.

[129] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *13th European Conference on Computer Vision*, pages 818–833, 2014.

[130] H. Zhang, M. Wang, R. Hong, and T. Chua. Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing. In *ACM Conference on Multimedia*, pages 781–790, 2016.

[131] P. Zhang, W. Zhang, W. Li, and M. Guo. Supervised hashing with latent factor models. In *SIGIR*, pages 173–182, 2014.

[132] X. Zhang, Y. Yang, Y. Zhang, H. Luan, J. Li, H. Zhang, and T. Chua. Enhancing video event recognition using automatically constructed semantic-visual knowledge base. *IEEE Trans. Multimedia*, 17(9):1562–1575, 2015.

[133] X. Zhang, H. Zhang, Y. Zhang, Y. Yang, M. Wang, H. Luan, J. Li, and T. Chua. Deep fusion of multiple semantic cues for complex event recognition. *IEEE Trans. Image Processing*, 25(3):1033–1046, 2016.