Part II

Learning Algorithms for Large-Scale Multimedia

4

Large-Scale Video Understanding with Limited Training Labels

Jingkuan Song, Xu Zhao, Lianli Gao and Liangliang Cao

4.1 Introduction

The challenge of video understanding lies in the gap between large-scale video data and the limited resource we can afford in both label collection and online computing stages. This chapter discusses both unsupervised and semi-supervised methods to facilitate video understanding tasks. In particular we consider two general research problems: video retrieval and video annotation/summarization. While the former focuses on designing efficient algorithms for retrieving useful information from large-scale data, the latter aims to reduce the size of the data. Specifically, video retrieval provides new models and methods for effectively and efficiently searching through the huge variety of video data that are available in different kinds of repositories (digital libraries, Web portals, social networks, multimedia databases, etc.). On the other hand, video annotation/summarization generates annotations or a short summary of a video. It provides efficient storage and quick browsing of a large collection of video data without losing important aspects. Depending on whether or not human labels are involved, video retrieval and annotation/summarization can be further categorized into unsupervised and semi-supervised based methods. In this work, we cover video retrieval and annotation/summarization, two approaches to fight the consequences of the big video data. We present the state-of-the-art research in each research field. Promising future trends on big video understanding are also proposed.

4.2 Video Retrieval with Hashing

4.2.1 Overview

With the rapid development of Internet techniques, video capturing devices, and video editing software, the number of online videos continues to grow at an astonishing speed. Meanwhile, video-related services, such as video sharing, monitoring, advertising, and recommendation, inspire online users' interests and participation in video-related activities, including uploading, downloading, commenting, and searching. A substantial number of videos are uploaded and shared on social websites every day.

Big Data Analytics for Large-Scale Multimedia Search, First Edition.
Edited by Stefanos Vrochidis, Benoit Huet, Edward Y. Chang, and Ioannis Kompatsiaris.

Companion website: www.wiley.com/go/vrochidis/bigdata

Content-based video hashing is a promising direction for content-based video retrieval, but it is more challenging than image hashing [1–4]. Essentially, the rich temporal information in videos is a barrier for directly utilizing image-based hashing methods [5]. A few attempts take a video as a set of images and ignore the temporal information [6–8]. Although the performance is promising, these methods learn hash codes for each frame where the structure of a video is not introduced.

Recently, deep learning has dramatically improved the state of the art in speech recognition, visual object detection and image classification (see chapter 1, ref. [11], [9, 10]). Inspired by this, researchers have started to apply deep features extracted from various deep convolutional neural networks (DCNNs) (e.g., VGG (see chapter 1, ref. [23]) and RestNet (see chapter 3, ref. [60]) to support video hashing. In general, they first conduct pooling on frame-level deep features to generate video-level features, and then project these high-dimensional features into low-dimensional hash codes [11]. Although pooling provides a solution for video hashing generation, it inevitably results in suboptimal binary codes, since video temporal information is significantly ignored. To capture the temporal information, the recurrent neural network (RNN) is used to achieve the state-of-the-art performance in video captioning [12, 13]. RNN-based hashing methods [14] utilize RNN and video labels to learn discriminative hash codes for videos. However, human labels are time- and labor-consuming, especially for large-scale datasets. More recently, Zhang et al. proposed self-supervised temporal hashing (SSTH) [15], which aims to improve video hashing by taking temporal information into consideration. They proposed a stacking strategy to integrate long short-term memory networks (LSTMs) [16] with hashing to simultaneously preserve a video's original temporal and spatial information using a short hash code. In this section, we introduce several video retrieval methods using hashing.

First, we integrate multiple features into hashing to improve the accuracy of hashing for unsupervised video retrieval. Most existing approaches use only a single feature to represent a video for retrieval. However, a single feature is often insufficient to characterize the video content. Besides, while the accuracy is the main concern in previous literature, the scalability of video retrieval algorithms for large-scale video datasets has been rarely addressed. In this section, we present a novel approach, multiple feature hashing (MFH), to tackle both the accuracy and the scalability issues of video retrieval. MFH preserves the local structure information of each individual feature and also globally considers the local structures for all the features to learn a group of hash functions which map the video keyframes into the Hamming space and generate a series of binary codes to represent the video dataset. We evaluate our approach on a public video dataset and a large-scale video dataset consisting of 132,647 videos, which was collected from YouTube by ourselves. The experiment results show that the proposed method outperforms the state-of-the-art techniques in accuracy.

Then, we propose a method called submodular video hashing (SVH), which can be either unsupervised or supervised. Unlike the previous methods, which capitalize on only one type of hash code for retrieval, SVH combines heterogeneous hash codes to effectively describe the diverse and multi-scale visual contents in videos. Our method integrates feature pooling and hashing in a single framework. In the pooling stage, we cast video frames into a set of pre-specified components, which capture a variety of semantics of video contents. In the hashing stage, we represent each video component as a compact hash code, and combine multiple hash codes into hash tables for effective search. To speed up the retrieval while retaining most informative codes, we propose a

graph-based influence maximization method to bridge the pooling and hashing stages. We show that the influence maximization problem is submodular, which allows a greedy optimization method to achieve a nearly optimal solution. Our method is extensively evaluated in both unsupervised and supervised scenarios, and the results on TRECVID multimedia event detection and columbia consumer video datasets demonstrate the success of our proposed technique.

4.2.2 Unsupervised Multiple Feature Hashing

4.2.2.1 Framework

The proposed framework based on MFH [17] for video retrieval is shown in Figure 4.1 and comprises two phases. In the first phase, which is offline, we use the proposed MFH algorithm to learn a series of s hash functions $\{h_1(\cdot), ..., h_s(\cdot)\}$, each of which generates one bit hash code for a keyframe according to the given multiple features. Each keyframe has s bits. Using the derived hash functions $\{h_1(\cdot), ..., h_s(\cdot)\}$, each keyframe for a dataset video can be represented by the generated s-sized hash codes in linear time. In the second phase, which is online, the query video's keyframes are also represented by s-sized hash codes mapped from the s hash functions. Video retrieval can be efficiently achieved where only efficient XOR operation on the hash codes is performed to compute the similarity between two videos.

The key research issue is how to train the hash functions which affect both accuracy and efficiency. In this section, we detail the proposed MFH algorithm. It is worthy noting that MFH is a general one which can be potentially applied to other large-scale search applications where the data are represented by multiple features.

4.2.2.2 The Objective Function of MFH

Let v be the number of features.[1] Given an integer $g \leq v$, $(x^g)_t \in \mathbb{R}^{d_g * 1}$ denotes the gth feature of tth training data, where d_g is the dimensionality of the gth feature. Suppose there are n training keyframes in total. $X^g = [(x^g)_1, (x^g)_2, ..., (x^g)_n] \in \mathbb{R}^{d_g * n}$ is the feature matrix corresponding to the gth feature of all the training data.

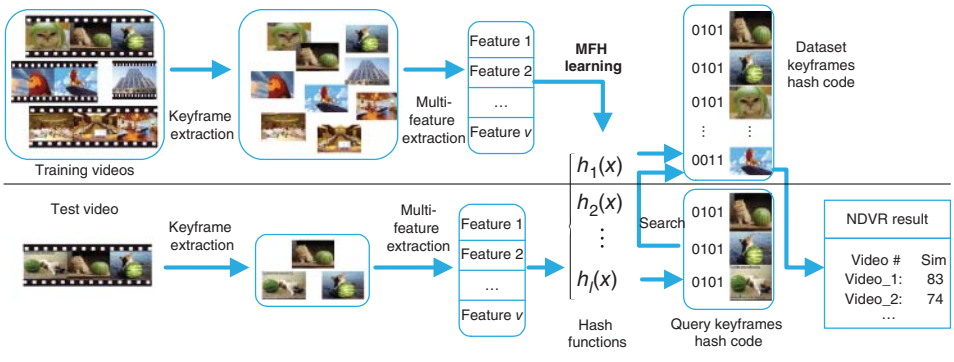


Figure 4.1 The proposed framework for video retrieval.

1 In our system, each keyframe is represented by the local feature Local binary pattern (LBP) and the global feature hue, saturation, value (HSV) color histogram. Therefore, $v = 2$ here. Yet, MFH is able to deal with more feature types when $v > 2$.

$x_t = [(x^1)_t^T, (x^2)_t^T, ..., (x^v)_t^T]^T \in \mathbb{R}^{d\times 1}$, where $d = \sum_{g=1}^{v} d_g$ is the vector representation of the tth training keyframe using all of the features and $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{d*n}$. MFH aims to learn a series of hash functions $\{h_1(\cdot), ..., h_s(\cdot)\}$, where s is the number of hash functions, i.e., the length of the hash code. Each of the s hash functions generates one bit hash code of the input keyframe. We further denote $Y = [(y_1)^T, ..., (y_n)^T]^T \in^{n\times s}$ as the hash codes of the training keyframes corresponding to all features and $Y^g = [(y_1^g)^T, ..., (y_n^g)^T]^T \in^{n\times s}$ as the hash codes of the training keyframes derived from the gth feature.

To exploit the individual structural information of each individual feature, we define v affinity matrices $A^g \in \mathbb{R}^{n\times n}$ $(1 \le g \le v)$, one for each feature as follows:

$$(A^g)_{pq} = \begin{cases} 1, \text{ if } (x^g)_p \in \mathcal{N}_k((x^g)_q) \text{ or } (x^g)_q \in \mathcal{N}_k((x^g)_p) \\ 0, \text{ else} \end{cases} \tag{4.1}$$

where $\mathcal{N}_k(\cdot)$ is the k-nearest-neighbor set and $1 \le (p, q) \le n$.

To learn the hash codes $(y^g)_t$ from the gth feature, a reasonable criterion is that similar items in the original space should have similar binary codes, which can be formulated as the following minimization problem:

$$\min_{Y^g} \sum_{p,q=1}^{n} (A^g)_{pq} \|(y^g)_p - (y^g)_q\|_F^2. \tag{4.2}$$

Given a training keyframe x_t, the overall hash codes y_t should be consistent with the hash codes $y_t^g|_{g=1}^v$ derived from each feature. In this way, the local structure information on each single feature is also globally considered and optimized. Therefore, we have the following minimization problem:

$$\min_{Y^g,Y} \sum_{g=1}^{v} \sum_{p,q=1}^{n} (A^g)_{pq} \|(y^g)_p - (y^g)_q\|_F^2 + \gamma \sum_{g=1}^{v} \sum_{t=1}^{n} \|y_t - (y^g)_t\|_F^2 \tag{4.3}$$

where γ is the parameter to balance the two parts. It is easy to see from Eq. 4.3 that the individual manifold structure of each feature type is preserved (the first term) and all the manifold structures are also globally considered in the optimization (the second term).

Recall that we intend to learn the hash codes of the training keyframes and a series of hash functions $\{h_1(\cdot), ..., h_s(\cdot)\}$ in a joint framework. We propose to simultaneously learn the hash codes of the training keyframes and the hash functions by minimizing the empirical error of the hash functions *w.r.t.* the learned hash codes Y. The proposed final objective function of MFH is given by:

$$\begin{aligned} \min_{Y,Y^g,W,b} \quad & \sum_{g=1}^{v} \sum_{p,q=1}^{n} (A^g)_{pq} \|(y^g)_p - (y^g)_q\|_F^2 + \gamma \sum_{g=1}^{v} \sum_{t=1}^{n} \|y_t - (y^g)_t\|_F^2 \\ & + \alpha(\|X^T W + \mathbf{1}b - Y\|_F^2 + \beta \|W\|_F^2) \\ \mathbf{s.t.} \quad & y_t \in \{-1, 1\}^s, (y^g)_t \in \{-1, 1\}^s, YY^T = I \end{aligned} \tag{4.4}$$

where α and β are the parameters, $\mathbf{1} \in \mathbb{R}^{n\times 1}$ is a column vector with all ones, and y_{tl} is lth bit hash code for x_t. In Eq. 4.4, $y_t \in \{-1, 1\}^s$ and $(y^g)_t \in \{-1, 1\}^s$ force the hash codes of y_t and $(y^g)_t$ to be binary codes. As we will show later on, the constraint $YY^T = I$ is imposed to avoid the trivial solution. For the simplicity of implementation for large-scale datasets, we adopt the linear transformation as the hash function. More specifically,

given a keyframe represented by its feature x_t, $w_l \in \mathbb{R}^{d\times 1}$ is the transformation matrix and $b_l \in \mathbb{R}$ is the bias term. $W = [w_1, w_2, \ldots, w_s] \in \mathbb{R}^{d\times s}$, $b = [b_1, b_2, \ldots, b_s] \in \mathbb{R}^{1\times s}$ and $\mathbf{1}$ is a vector of all ones.

4.2.2.3 Solution of MFH

In this subsection, we detail the approach to optimize the objective function of MFH. By setting the derivative of Eq. 4.4 *w.r.t.* b and W to zero, we get:

$$b = \frac{1}{n}(\mathbf{1}^T Y - \mathbf{1}^T X^T W),\ W = (XL_cX^T + \beta I)^{-1}XL_cY, \tag{4.5}$$

where $L_c = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is the centering matrix. Note that $L_c = (L_c)^T = L_c(L_c)^T$. Replacing W and b by Eq. 4.5, we have

$$\begin{aligned} X^T W + \mathbf{1}b - Y &= X^T W + \mathbf{1}\frac{1}{n}(\mathbf{1}^T Y - \mathbf{1}^T X^T W) - Y \\ &= L_c X^T W - L_c Y = L_c X^T (XL_c X^T + \beta I)^{-1} XL_c Y - L_c Y \end{aligned} \tag{4.6}$$

Let $M = (XL_cX^T + \beta I)^{-1}$. $\|X^T W + \mathbf{1}b - Y\|_F^2 + \beta\|W\|_F^2$ can be rewritten as:

$$\|X^T W + \mathbf{1}b - Y\|_F^2 + \beta\|W\|_F^2 = trY^T BY,\ B = (L_c - L_c X^T (XL_c X^T + \beta I)^{-1} XL_c) \tag{4.7}$$

Meanwhile, we have

$$\sum_{p,q=1}^{n} (A^g)_{pq}\|(y^g)_p - (y^g)_q\|_F^2 = tr\ ((Y^g)^T L^g Y^g) \tag{4.8}$$

where $L^g = N^g - A^g$ is the Laplacian matrix of the gth feature and N^g is the diagonal matrix with its diagonal element $N_{ii}^g = \sum_j A_{ij}^g$. Note that $\sum_{g=1}^{v}\sum_{t=1}^{n}\|y_t - (y^g)_t\|_F^2$ can also be written as $\sum_{g=1}^{v}\|Y - Y^g\|_F^2$. Then the objective function shown in Eq. 4.4 becomes

$$\begin{aligned} \min_{Y,Y^g}\quad & \sum_{g=1}^{v}(tr((Y^g)^T L^g Y^g) + \gamma\|Y - Y^g\|_F^2) + \alpha tr(Y^T BY) \\ \mathbf{s.t.}\quad & Y^T Y = I \end{aligned} \tag{4.9}$$

Setting the derivative of Eq. 4.9 *w.r.t.* Y^g to be zero, we have

$$2L^g Y^g - 2\gamma(Y - Y^g) = 0 \Rightarrow Y^g = \gamma(L^g + \gamma I)^{-1} Y \tag{4.10}$$

Let $C^g = \gamma(L^g + \gamma I)^{-1}$. Note that $C^g = (C^g)^T$. Then we arrive at

$$Y^g = C^g Y \tag{4.11}$$

Replacing Y^g with Eq. 4.11, the objective function in Eq. 4.9 becomes:

$$\min_{YY^T = I} tr(Y^T DY) \tag{4.12}$$

where D is defined as

$$D = \sum_{g=1}^{v}(C^g L^g C^g + \gamma(I - C^g)^2) + \alpha B = \gamma\sum_{g=1}^{v}(I - C^g) + \alpha B \tag{4.13}$$

Algorithm 4.1 The MFH algorithm.

```
for g = 1 to v do
    for p = 1 to n do
        for q = 1 to n do
            Compute L^g_{p,q} according to Eq. 4.8;
        end for
    end for
end for
Compute B according to Eq. 4.7;

Compute D according to Eq. 4.13;
Compute Y by performing eigenvalue decomposition on D.
Output W and b according to Eq. 4.5.
```

The optimization problem shown in Eq. 4.12 can be solved by performing the eigen-value decomposition of D. Y can be obtained by the s eigenvectors of D corresponding to the s smallest eigenvalues.[2] In summary, the training processing of the proposed MFH algorithm is listed as follows.

Once W and b are obtained, they can be readily used to generate the hash codes of database keyframes as well as query keyframes. For example, given a keyframe x_t, we first compute its relaxed hash code y_t in continuous domain by $y_t = W^T x_t + b$. Then, we binarize it by comparing each dimension of y_t with its median of all dimensions. If it is greater than the median, we set it as 1. Otherwise, we set it as -1. To represent a video clip, Wu et al. proposed averaging the visual feature of all its keyframes as the representation of the whole video clip [18]. Following Wu's way, we can also generate the hash codes for a video clip by averaging the relaxed codes of all its keyframes and then binarizing them. As a result, each video clip is represented by a single s-bit hash code. In this work, we use the above video representation to avoid pairwise keyframe comparisons. The number of common bits along all dimensions shared by two videos is approximated as the video similarity.

4.2.2.3.1 Complexity Analysis

In the previous subsections we proposed a multi-feature hashing algorithm MFH for near-duplicate video retrieval (NDVR). During the training phase, to compute the matrix A_1^g, we need to perform a k-nearest-neighbors search for each feature type whose time complexity is $O(k \times n^2)$. Since we have v feature types, the time complexity is $O(v \times k \times n^2)$. Because $v \ll n$ and $k \ll n$, the time complexity for computing A_1^g is $O(n^2)$. To compute the matrix A_2^g, we need to search the n training data points for each data point whose time complexity is $O(n^2)$. Also, we need to perform an eigenvalue decomposition to get Y, whose time complexity is n^3. Then, we need to solve a linear system when computing W and b, whose time complexity is $O(n^2)$ approximately. Therefore, the time complexity for the training of MFH is $O(n^3)$ approximately. Once the hash functions, i.e., W and b, are obtained, we only need to compute the hash code

2 The trivial solution corresponding to the eigenvalue of zero is removed.

to map the whole database into the Hamming space with a linear time complexity. All of the above steps are done offline.

4.2.3 Submodular Video Hashing

4.2.3.1 Framework

The proposed framework based on SVH for video retrieval is shown in Figure 4.2, which comprises two phases. In the first phase, which is the pooling stage, we cast video frames into a set of pre-specified components, which capture a variety of semantics of video contents. In the second stage, which is the hashing stage, we represent each video component as a compact hash code, and combine multiple hash codes into hash tables for effective search.

4.2.3.2 Video Pooling

The basic idea of our hashing algorithm is to exploit the rich semantics of videos and construct binary codes corresponding to different characteristics of video contents. Our methods contrast with previous image-based hashing methods by considering video feature pooling in frame-level. Suppose there is a video with T frames, where each frame is represented by a feature vector x_t. We generate the temporal pooling component k as:

$$z^k = \sum_{t=1}^{T} p_t^k x^t \tag{4.14}$$

Here p_t^k stands for the probability $Pr(k|x_t)$ that captures the amount of contribution of frame t in forming the scene component k. We approximate the probability using soft

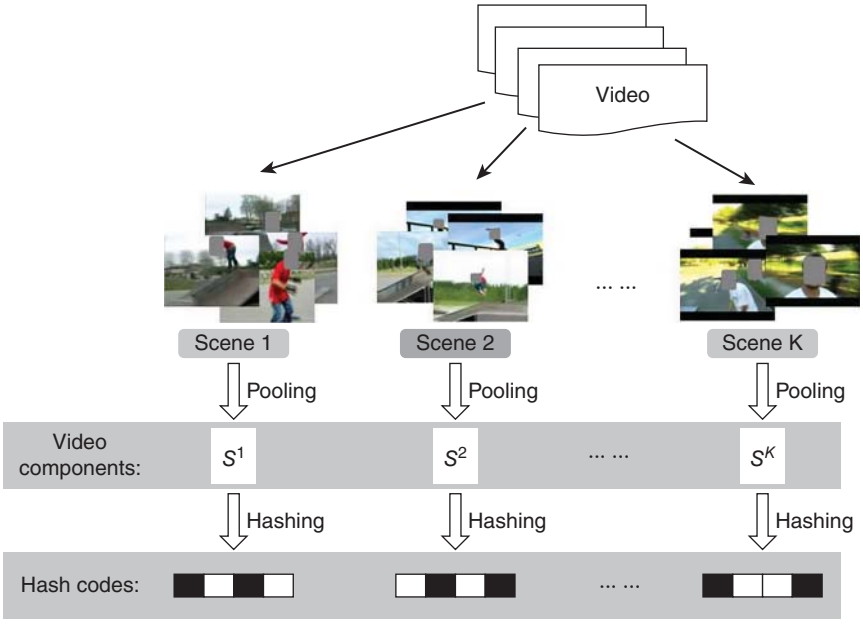


Figure 4.2 The proposed framework of SVH for video retrieval.

vector quantization on GIST features. From Eq. 4.14, we can see that to obtain different scene components $z^k = \sum_{t=1}^{T} p_t^k x^t$, in fact is equivalent to decomposing each frame feature x_t into different scenes

$$x_t \rightarrow [p_t^1 x_t, p_t^2 x_t, ...], \tag{4.15}$$

From Eqs 4.14 and 4.15 one can see that our new model is a feature pooling method [19]. The unique characteristic of our method is that our pooling weights p_t^k are on video frames instead of the spatial domain. Here we force the pooling weights p_t^k to follow the constraint $\sum_k p_t^k = 1$. We use GIST features to represent the scenes of video frames and then to compute p_t^k. Note that the pooling weights are computed based on Euclidean distance, which is not reliable for sparse histogram features like scale-invariant feature transform (SIFT). On the other hand, GIST is easy to compute and gives a reliable measure of how different two frames look holistically (see chapter 1, ref. [29]). With all the training data, we compute the GIST features for all frames and cluster them into C centers using the K-means algorithm where the number of C is empirically set. The C center is represented as $g_c^1, g_c^2, ..., g_c^C$. To compute the pooling weight p_t^k for frame t in a training or test video clip, we extract it GIST feature vector g_t. The pooling weight is derived by comparing g_c^k and g_t. One simple way to compute pooling weights is by vector quantization (VQ), which forces all but the nearest p_t^k to be zero. However, VQ is well-known to be sensitive to noises. Here, we consider the soft voting pooling weights as

$$p_t^k = \frac{1/(d_t^k)^3}{\sum_{l=1}^{C} 1/(d_t^l)^3} \tag{4.16}$$

where d_t^k is the Euclidean distance between centers g_c^k and frame feature g_t. It is not difficult to see that our pooling weights satisfy the constraint $\sum_k p_t^k = 1$.

4.2.3.3 Submodular Video Hashing

Suppose we have nf features indexed by $f = 1, ..., nf$, where each (f) is captured by multiple (C) components $(v_f^1, ..., v_f^C)$ and each component (v_f^j) is expressed by a compact binary hash code1. Our first observation is that not every code is equally informative and there can be significant redundancy among the codes. On the other hand, it is critical to use compact representation for large-scale video retrieval. Inspired by these observations, we aim to select a small number of "informative" codes which can well "represent" the entire set of codes. We found these concepts can be effectively modeled using a graph over the codes whose edge weights capture the degree of overlapping or similarity between pairwise codes. In this section, we assume that the mechanism to compute pairwise code similarity is known (which will be addressed in the next subsection).

Let $\mathbb{G} = \{\mathbb{V}, \mathbf{W}\}$ be a graph with node set $\mathbb{V}$ consisting of the hash codes considered, i.e., $\mathbb{V} = \{v_f^k | k = 1, ..., C; f = 1, ..., n_f\}$, and similarity matrix $\mathbf{W} = [w_{ij}]$, where w_{ij} captures the similarity between codes i and j. Our goal is to select m representative nodes from the graph, where m can be manually specified or determined automatically. We cast this node selection problem as an influence maximization problem where the influence of the selected m nodes can be maximally propagated to the rest of the nodes in the graph.

Denote $\mathbb{A}$ as the set of selected nodes whose influences are assigned and fixed to 1's. We use a score $u_{\mathbb{A}}(i) \in [0, 1]$ to quantify the influence node i received from $\mathbb{A}$. For technical reason, that will be clear soon, we introduce a sink node s to the graph that is

connected to each node with a small constant weight. The sink node s is very "cool" in that it is never influenced by others or tries to influence others. So its influence is fixed to 0. For simplicity and a little abuse of notations, we still denote the graph as $\mathbb{G} = \{\mathbb{V}, \mathbf{W}\}$ but keep in mind that $s \in \mathbb{V}$ and $\mathbf{W}$ is expanded accordingly. We also denote $\hat{\mathbb{A}} = \mathbb{A} \cup s$ and $\mathbb{A}$ as the set of remaining nodes, i.e., $\hat{\mathbb{A}} \cap \mathbb{N} = \emptyset$ and $\hat{\mathbb{A}} \cap \mathbb{N} = \mathbb{V}$. Formally, we propose the following influence maximization framework:

$$\max_{\mathbb{A} \subset \mathbb{V}} \Omega(\mathbb{A}) := \sum_{i \in \mathbb{N}} u_{\mathbb{A}}(i) \tag{4.17}$$

To instantiate the above framework, a concrete model for influence propagation is needed. Specifically, we use the concept of harmonic fields from Zhu et al. [20], which has been shown to be a highly effective label propagation model. Denote $u_{\mathbb{A}}(i) = 1$ if $i \in \mathbb{A}$ and $u_{\mathbb{A}}(s) = 0$. Our influence model enjoys a "harmonic" interaction among the nodes as follows:

$$u_{\mathbb{A}}(i) = \frac{1}{d_i} \sum_{j \in \mathbb{N}(i)} w_{ij} u_{\mathbb{A}}(j), i \in \mathbb{N} \tag{4.18}$$

where $\mathbb{N}(i)$ denotes the set of neighbors of node i and $d_i = \sum_{j \in \mathbb{N}(i)} w_{ij}$ is the degree of node i. In other words, the influence on node i is the weighted sum of the influences on its neighbors.

Since our objective function (4.17) for the hash function selection problem is submodular [8], we can obtain a nearly optimal combination of hash codes by a greedy algorithm. We start from an empty set $A_0 := \emptyset$ and iteratively add a code y to $\mathbb{A}_i$ that maximizes the increase in influence of Ω. Specifically, we select the node y_{i+1} in step $i + 1$ using the following criterion:

$$y_{i+1} = \arg\max_{y \in \mathbb{N}_i} \Omega(\mathbb{A}_i \cup \{y\}) - \Omega(\mathbb{A}_i) \tag{4.19}$$

4.2.4 Experiments

We experimentally evaluated the performance of MFH and SVH and compared it with existing state-of-the-art methods.

4.2.4.1 Experiment Settings

4.2.4.1.1 Video Datasets

We used two datasets in the experiments.

The CC_WEB_VIDEO Dataset [21] is provided by City University of Hong Kong and Carnegie Mellon University. It consists of 24 sets of video clips (13,129 video clips in total) downloaded from video-sharing websites.

The Combined Dataset is a larger video dataset created by ourselves by adding CC_WEB_VIDEO to our video dataset downloaded from YouTube.

4.2.4.1.2 Visual Features

In the CC_WEB_VIDEO dataset, a shot-based sampling method was used to extract a total of 398,078 keyframes. These keyframes are provided in the dataset. We further extracted the LBP feature on the keyframes as a local feature, which will be used together with the existing global feature HSV provided by the dataset.

In the Combined Dataset, we firstly detected the shots from which the keyframes were extracted. Then we extracted two features for each keyframe, HSV and LBP, to be used as a global feature and a local feature, respectively. HSV is a global color histogram with 162 dimensions and LBP is a local content feature with 256 dimensions.

4.2.4.1.3 Algorithms for Comparison

To evaluate video retrieval accuracy and efficiency, we present an extensive comparison of the proposed method with a set of existing video retrieval methods, which are briefly described as follows.

We use the global color histogram method as a baseline method. Each keyframe is represented as a color histogram feature vector. Each video is finally defined as a feature vector of a normalized color histogram over all the keyframes in the video. We also compared the proposed method with the local feature based method [18], the spatiotemporal feature based method [22], self-taught hashing [23] and spectral hashing [24].

Following the work in [22], we also adopted the mean average precision (MAP) metric to evaluate the performance. We further used a precision-recall curve to evaluate the performance on both datasets.

4.2.4.2 Results

4.2.4.2.1 CC_WEB_VIDEO

We first tested different methods on the CC_WEB_VIDEO dataset. The same 24 queries as in [18, 22] were used. The MAP results of different methods can be seen in Table 4.1.

From Table 4.1, we can see that the proposed MFH achieves the best results on MAP, although the margins are minor. This is probably because the dataset is not big enough and thus less noise can be introduced for most methods. The performance of different methods cannot be fully explored. Next, we focus on the large dataset.

4.2.4.2.2 Combined Dataset

To further test the accuracy and scalability of MFH, we tested MFH on the Combined Dataset and compare it with existing methods. More specifically, we used the same 24 queries and ground truth as in [18, 22] to search the whole Combined Dataset. Several parameters need to be tuned for some of the algorithms. For each parameter we have a set of candidate values and we iteratively combined the parameters to form various parameter settings. For each parameter combination we ran the same experiment to evaluate its impact on the performance. Finally, the parameter setting which results in the best performance was selected for each method. For the parameters in MFH, SPH and STH, including γ,α,β, we tuned them from $10^{-6}, 10^{-3}, 10^{0}, 10^{3}, 10^{6}$ and tuned the

Table 4.1 Comparison of MAP on CC_WEB_VIDEO.

Methods	GF	LF	ST_lbp	ST_ce	MFH
MAP	0.892	0.952	0.953	0.950	0.954

GF, global feature; LF, ST_lbp, ST_ce, MFH, see [18] and [22].

Table 4.2 Comparison of MAP and time for the Combined Dataset.

Methods	MAP	Time(s)
SPH	0.5941	0.4907
GF	0.6466	1.3917
STH	0.7536	0.6439
MFH_lbp	0.7526	0.6445
MFH_hsv	0.8042	0.4508
MFH	0.8656	0.5533

GF, global feature; SPH, (see chapter 9, ref. [24]); STH, see [23]; MFH, see [7].

length of the binary code from 200 to 400. On this large Combined Dataset, we cannot compare with the local feature based method LF proposed in [18] because this method utilizes interest points matching to measure the similarity of two keyframes. This is impractical in such a large dataset with 2,570,554 keyframes. We were also unable to extract the spatiotemporal features ST_lbp and ST_ce because it would take an extremely long time for our computer to extract the features from 132,647 videos. As a result, only the GF, STH, SPH, MFH_hsv, MFH_lbp, and MFH methods are compared.

The results for MAP and search time are shown in Table 4.2. We have the following observations:

- MFH achieves the highest MAP and outperforms existing methods by a large margin. MFH can have more than 86% MAP, while SPH has less than 60% MAP.
- SPH, STH, and MFH all construct binary codes for the videos and perform the search using the Hamming space. Even in the Matlab environment they can search the large dataset within 1 second. However, SPH's performance is really unsatisfactory probably because the video dataset is not in accordance with the assumption that all the data points are uniformly distributed in a hyper-rectangle in high-dimensional space. This also shows that MFH is more capable of preserving local data distribution information.
- GF performs the search using the Euclidean distance and its performance is not good. Its accuracy is much worse than that of MFH. Furthermore, its search time is two to three times longer than that of the hashing methods.
- MFH outperforms MFH_lbp and MFH_hsv in general, which demonstrates the effectiveness of combining multiple features in video retrieval.

4.2.4.3 Evaluation of SVH

We also experimentally evaluated the performance of SVH and compare it with existing state-of-the-art methods on two datasets.

The TRECVID Multimedia Event Detection (MED 2011) dataset is a large annotated dataset specifically designed to model complex video events. The evaluation of MED is separated into two test sets: the mid-size dryrun evaluation and the large final evaluation set.

The Columbia Consumer Video (CCV) dataset [25] contains 9317 YouTube videos over 20 semantic categories.

We applied the unsupervised measure for the MED dataset and the semi-supervised measure for CCV dataset. Since each video in the CCV dataset is assigned a label from 20 semantic categories, we evaluated retrieval performance for all categories.

4.2.4.3.1 Results

MED dataset The results in Figure 4.3a show that our submodular hashing method is significantly better. This confirms the crucial benefit of employing multiple scene components for video representation and the submodular selection strategy for hash table construction The results also show that the performance of low-level features such as LBP and color histogram are inferior to SIFT histogram, which motivated us to focus on powerful features like SIFT and its variants in the final run.

The results for the final evaluation dataset are shown in Figure 4.3b, which shows that semantic feature vectors work better than SIFT features for video retrieval, but our submodular hash methods still clearly boost the performance over these features. The results also show that random projection works better than spectral hashing, which is probably because the objective function of spectral hashing applies to the Hamming distances between all the samples, of which a large proportion has been ignored in our hash table structure. Despite that, the contributions of the video component and submodular selection are consistent in both cases.

In this dataset, we employed three features: sparse sift, dense sift, and pyramid histogram of oriented gradients (PHOG). Since the similarity of different hash codes depends on the category label, we also compared the retrieval performance for each category. To make a fair comparison, we report the results using four hash tables. The average recall is used to measure retrieval accuracy, as we did on the MED dataset. As shown in Figure 4.4, our submodularity based video hashing is consistently better than hashing accumulated video features.

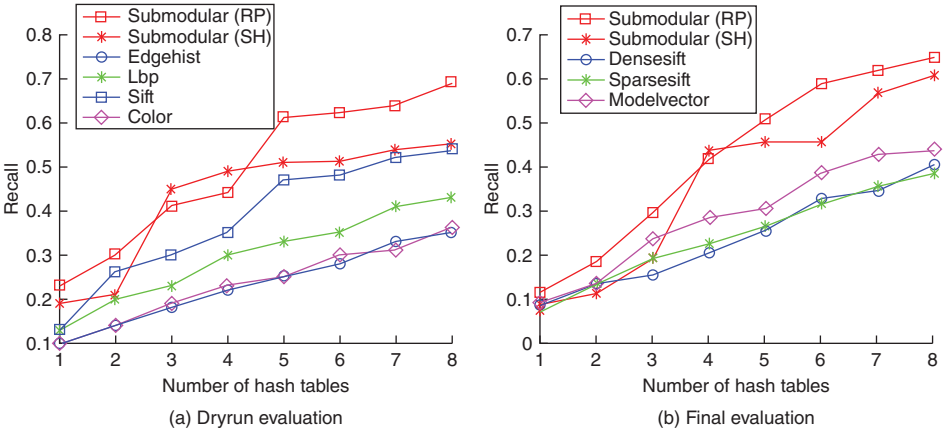


Figure 4.3 Comparison of recall on MED dataset.

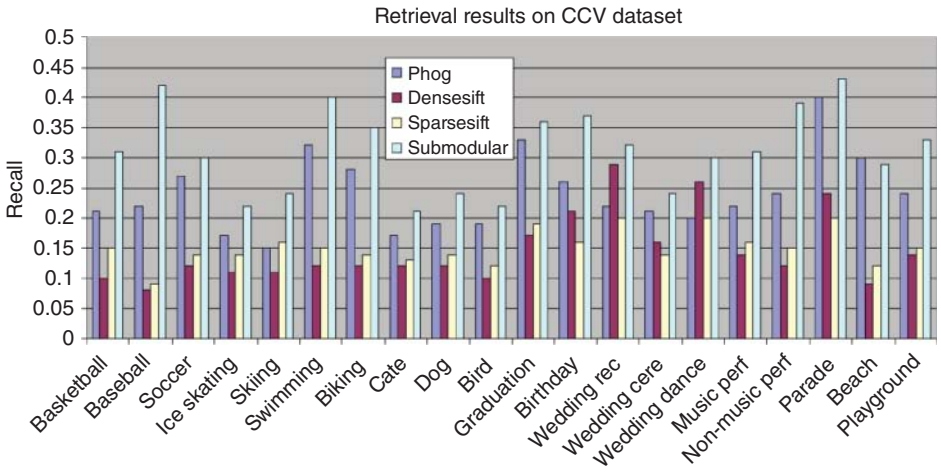


Figure 4.4 Video retrieval performance on the CCV dataset.

4.3 Graph-Based Model for Video Understanding

4.3.1 Overview

Recently, we have witnessed an exponential growth of user-generated videos and images due to the booming use of social networks such as Facebook and Flickr. There are also some well-known public datasets for large-scale video understanding [26]. Consequently, there are increasing demands to effectively organize and access these multimedia data via annotation/summarization. While video retrieval provides methods for effectively and efficiently searching through the huge variety of video data that are available in different kinds of repositories, video annotation/summarization/captioning generates annotations or a short summary of a video. It provides efficient storage and quick browsing of large collection of video data without losing important aspects.

Graph-based learning is an efficient approach for modeling data in various machine learning schemes, i.e., unsupervised learning [27, 28], supervised learning [29], and semi-supervised learning (SSL) [28, 30]. An important advantage of working with a graph structure is its ability to naturally incorporate diverse types of information and measurements, such as the relationship between unlabeled data, labeled data, or both labeled and unlabeled data.

By exploiting a large number of unlabeled data with reasonable assumptions, SSL can reduce the need for expensive labeled data and thus achieve promising results, especially for noisy labels. The harmonic function approach [20] and local and global consistency (LGC) [31] are two representative graph-based SSL methods. The harmonic function approach [20] emphasizes the harmonic nature of the energy function and LGC considers the spread of label information in an iterative way. While these two methods are transductive, manifold regularization (MR) [32] is inductive. In practice, MR extends regression and SVM to SSL methods such as Laplacian regularized least

squares (LapRLS) and Laplacian support vector machines (LapSVM), respectively. By adding a geometrically based regularization term [33]. In [34], a so-called correlation component manifold space learning (CCMSL) was developed to learn a common feature space by capturing the correlations between heterogeneous databases. In [35], a content similarity based fast reference frame selection algorithm was proposed for reducing the computational complexity of the multiple reference frames based inter-frame prediction.

First, we propose to annotate videos in a semi-supervised way. Specifically, we propose learning an optimal graph (OGL) from multi-cues (i.e., partial tags and multiple features), which can more accurately encode the relationships between data points. Then, we will incorporate the learned optimal graph with the SSL model and further extend this model to address out-of-sample extension and noisy label issues.

We specifically address the problem of annotating actions in videos using a graph-based model. We present a context-associative approach to recognize activity with human-object interaction. The proposed system can recognize incoming visual content based on the previous experienced activities.

Finally, we address the problem of video summarization. We have developed a novel approach, termed *event video mashup* (EVM), to automatically generate a unified short video from a collection of Web videos to describe the storyline of an event. To advance research on animated GIF understanding, we collected a new dataset, Tumblr GIF (TGIF), with 100K animated GIFs from Tumblr and 120K natural language descriptions obtained via crowdsourcing.

4.3.2 Optimized Graph Learning for Video Annotation

4.3.2.1 Framework

In this section we introduce our OGL method [36], which consists of two phases (see Figure 4.5). First, a similarity graph is constructed on each feature (multiple feature graph) and also on the partial tags (partial label graph) to exploit the relationship among the data points. Partial tags means that tags are provided only for a part of the training data. Then, optimal graph learning is applied to these graphs to construct an optimal graph, which is integrated with SSL for the task of image and video annotation. Note that in theory our approach can be integrated with all kinds of graph-based algorithms.

4.3.2.2 OGL

4.3.2.2.1 Terms and Notations

We first introduce the notations which will be used in the rest of the section. $X = \{x_1, x_2, ..., x_n\}$ represents a set of n images, $y_i = \{0, 1\}^c$ is the label for the ith image ($1 \le i \le n$), and c is the number of annotations/classes. The first l points x_i ($i \le l$) are labeled as Y_l, and the remaining u points x_i ($l + 1 \le i \le n$) are unlabeled. The goal of transductive graph-based SSL is to predict the label F_u of the unlabeled points. Define a $n \times c$ matrix $F = \begin{bmatrix} F_l \\ F_u \end{bmatrix}$ with $F_l = Y_l$ and $F_u = \{0\}^{u*c}$.

Suppose that for each image, we have v features. Let $X^t = \{x_i^t\}_{i=1}^n$ denote the feature matrix of the tth view of training images, where $t \in \{1, ..., v\}$.

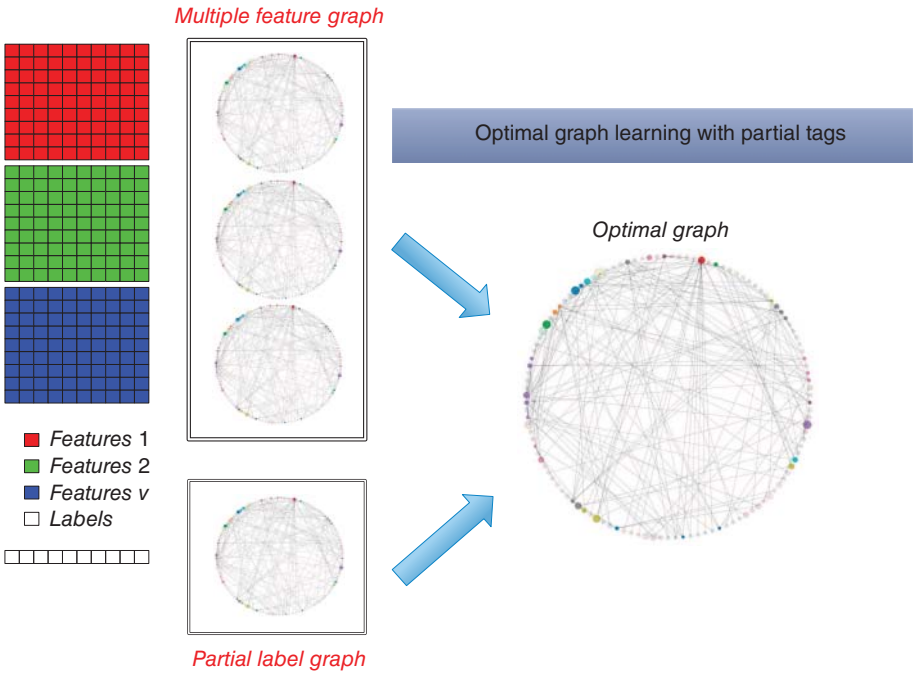


Figure 4.5 The overview of OGL.

4.3.2.2.2 *Optimal Graph-Based SSL*

The traditional graph-based SSL usually solves the following problem:

$$\min_{F,F_l=Y_l} \sum_{ij} \|f_i - f_j\|_2^2 s_{ij} \tag{4.20}$$

where f_i and f_j are the labels for the ith and jth images, and S is the affinity graph with each entry s_{ij} representing the similarity between two images. The affinity graph $S \in \mathbb{R}^{n\times n}$ is usually defined as follows:

$$s_{ij} = \begin{cases} e^{-\|x_i - x_j\|_2^2/2\sigma^2}, & \text{if } x_i \in \mathcal{N}_K(x_j) \text{ or } x_j \in \mathcal{N}_K(x_i) \\ 0, & \text{else} \end{cases} \tag{4.21}$$

where $\mathcal{N}_K(\cdot)$ is the K nearest-neighbor set and $1 \le (i,j) \le n$. The variance σ will affect the performance significantly and is usually empirically tuned. Also, the similarity graph is often derived from a single information cue. To address these issues, we propose to learn an optimal graph S from multiple cues.

The multiple cues include the given label information F and the multiple features $X^t = \{x_i^t\}_{i=1}^n$. An optimal graph S should be smooth on all these information cues, which can be formulated as:

$$\min_{S,\alpha} g\ (F,S) + \mu \sum_{t=1}^{v} \alpha^t h\ (X^t,S) + \beta r\ (S,\alpha) \tag{4.22}$$

where $g\ (F, S)$ is the penalty function to measure the smoothness of S on the label information F and $h\ (X^t, S)$ is the loss function to measure the smoothness of S on the feature X^t. $r\ (S, \alpha)$ are regularizers defined on the target S and α. μ and β are balancing parameters, and α^t determines the importance of each feature.

The penalty function $g(F, S)$ should be defined in such a way that close labels have high similarity and vice versa. In this section, we define it as follows:

$$g(F, S) = \sum_{ij} \|f_i - f_j\|_2^2 s_{ij} \tag{4.23}$$

where f_i and f_j are the labels of data points x_i and x_j. Similarly, $h(X^t, S)$ can be defined as:

$$h(X^t, S) = \sum_{ij} \|x_i^t - x_j^t\|_2^2 s_{ij} \tag{4.24}$$

Note that for simplicity we use the distance-based method to learn the similarity graph here. Other options based on reconstruction coefficients methods can be utilized to achieve better performance, which is discussed in the next section. Instead of preserving all the pairwise distances, we consider preserving the pair distances of the K nearest-neighbors here, i.e., if x_i^t and x_j^t (or f_i and f_j) are not K nearest-neighbors of each other, their distance will be set to a large constant. The regularizer term $r(S, \alpha)$ is defined as:

$$r(S, \alpha) = \frac{\mu\gamma}{\beta} \|S\|_F^2 + \|\alpha\|_2^2 \tag{4.25}$$

We further constrain that $S \geq 0$, $S\mathbf{1} = \mathbf{1}$, $\alpha \geq 0$ and $\alpha^T \mathbf{1} = \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{N\times 1}$ is a column vector with all ones. Then we can obtain the objective function for learning the optimal graph by replacing $g(F, S)$, $h(X^t, S)$ and $r(S, \alpha)$ in Eq. 4.22 using Eqs. 4.23, 4.24 and 4.25. By combining Eq. 4.20 with Eq. 4.22, we can obtain the objective function for optimal-graph based SSL, as follows:

$$\begin{aligned} &\min_{S,F,\alpha} \sum_{ij} \|f_i - f_j\|_2^2 s_{ij} + \beta\|\alpha\|_2^2 + \mu \left(\sum_{tij} (\alpha_t \|x_i^t - x_j^t\|_2^2 s_{ij}) + \gamma \|S\|_F^2 \right) \\ &s.t. \left\{ S \geq 0, S1 = 1, F_l = Y_l, \alpha \geq 0, \alpha^T 1 = 1 \right. \end{aligned} \tag{4.26}$$

4.3.2.2.3 Iterative Optimization

We propose an iterative method to minimize the above objective function in Eq. 4.26. First, we initialize $S = \sum_t S^t / v$ with each S^t being calculated using Eq. 4.21, and we initialize $\alpha^t = 1/v$. We further normalize S as $S = (D^{1/2})^T S D^{1/2}$. Once these initial values are given, in each iteration, we update F given S and α, and then update S and α by fixing the other parameters. These steps are described below.

Update *F* By fixing S and α, we can obtain F by optimizing Eq. 4.26. This is equivalent to optimizing the following objective function:

$$\min_{F, F_l = Y_l} \sum_{ij} \|f_i - f_j\|_2^2 s_{ij} = \min_{F, F_l = Y_l} \|F(I - S)F^T\|_F^2 \tag{4.27}$$

where I is an identity matrix. Let $L = I - S$, and differentiate the objective function in Eq. 4.27 with respect to F, to obtain:

$$LF = 0 \Rightarrow \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix} \begin{bmatrix} F_l \\ F_u \end{bmatrix} = 0, \Rightarrow L_{ll}F_l + L_{lu}F_u = 0, \; L_{ul}F_l + L_{uu}F_u = 0 \tag{4.28}$$

Then we can obtain:

$$F_u^* = -{L_{uu}}^{-1}L_{ul}F_l \tag{4.29}$$

Update S By fixing F and α, we can obtain S by optimizing Eq. 4.26. This is equivalent to optimizing the following objective function:

$$\sum_{ij} \|f_i - f_j\|_2^2 s_{ij} + \mu \sum_{tij} (\alpha_t \|x_i^t - x_j^t\|_2^2 s_{ij}) + \mu\gamma \|S\|_F^2 \tag{4.30}$$

and it is equivalent to:

$$\min_{S, S\geq 0, S1=1} \sum_i \left\| s_i + \frac{a_i + \mu b_i}{2\mu\gamma} \right\|_2^2 \tag{4.31}$$

where $b_i = \{b_{ij}, 1 \leq j \leq n\}$ with $b_{ij} = \sum_t \alpha_t \|x_i^t - x_j^t\|_2^2$ and $a_i = \{a_{ij}, 1 \leq j \leq n\} \in R^{1\times n}$ with $a_{ij} = \|y_i - y_j\|_2^2$.

The problem in Eq. 4.31 is simplex and we use the accelerated projected gradient method to linearly solve it. The critical step of the projected gradient method is to solve the following proximal problem:

$$\min_{x\geq 0, x^T 1=1} \frac{1}{2}\|x - c\|_2^2 \tag{4.32}$$

This proximal problem can be solved using the Karush-Kuhn-Tucker approach. Then each s_i can be efficiently solved, and we can get the updated graph S.

Update α By fixing F and S, we can obtain α by optimizing Eq. 4.26. This is equivalent to optimizing the following objective function:

$$\min_{\alpha\geq 0, \alpha^T 1=1} \mu \sum_t \alpha_t \left(\sum_{ij} \|x_i^t - x_j^t\|_2^2 s_{ij} \right) + \beta \|\alpha\|_2^2, \Rightarrow \min_{\alpha\geq 0, \alpha^T 1=1} \mu d\alpha + \beta \|\alpha\|_2^2 \tag{4.33}$$

where $d = \{d_t, 1 \leq t \leq v\}$ with $d_t = \sum_{ij} \|x_i^t - x_j^t\|_2^2 s_{ij}$. It can be reformulated in the form of the problem in Eq. 4.32 and can be solved similarly to obtain α.

We update F, S and α iteratively until the objective function Eq. 4.26 converges.

4.3.3 Context Association Model for Action Recognition

While the previous section discusses video annotation, this section focuses on action recognition in videos. It is inspired by the context memory model (CMM) [37] and we can improve CMM to explore video sequences and organize the episodic information for recognition by recollection. In this section, we describe two different memory units in CMM, and propose our framework [38] based on them. The data in memory are processed and maintained under this framework.

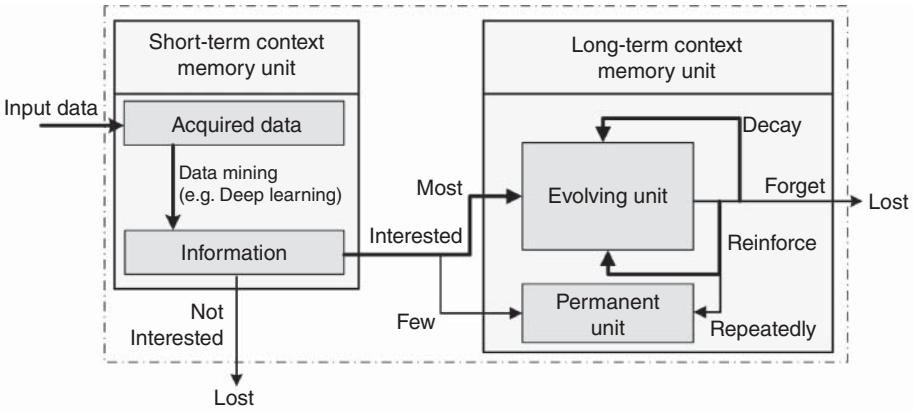


Figure 4.6 Framework of our context memory. The sensory memory is presented for acquiring data and incorporated into the short-term context memory.

4.3.3.1 Context Memory

There are two different memory units in CMM: short-term context memory (SCM) and long-term context memory (LCM) units, as shown in Figure 4.6. The sensory memory [39] is mainly used for extracting information from input data, but is omitted in CMM. We incorporated the data mining part as the sensory model into SCM for acquiring representations. Because permanent memory is rare, we focus on SCM and *evolving units*.

In this work, data acquiring, processing, and maintaining are incorporated into one framework, as shown in Figure 4.6. SCM acquires and processes data to find meaningful and useful information. Many algorithms fit into SCM, such as those in human detection, object tracking, action recognition, and so on. Only the messages of interest from the obtained information will be structured and stored in LCM, and these are used for retrieval. The data in LCM are always under memory evolution, where some can decay or be forgotten, and some can be reinforced.

Hierarchical structure The evolving unit is a key part. Its context structure organizes the detected context elements into an entity based on the relationships. Figure 4.7 shows the summary of the hierarchy.

The basic element of context memory is *attribute* (e.g., an object, a person). An *instance* is formed by several related attributes (e.g., a person is moving a box). Several temporally consecutive instances can form a *cluster* (e.g., an event "arrange objects" consists of "a person is reaching a box", "a person is moving a box", "a person is putting a box" and so on). At a certain time, the memory comprises many clusters, and we call this a *snapshot*, which is used for memory evolution. The context memory is dynamic along the timeline, which means that the data in the memory can decay, be reinforced, or be forgotten.

Temporal relations Only the static relations of attributes and instances are modeled in CMM, where the context cluster is defined as a group of a set of context instances with the similarity over a threshold, as shown in Figure 4.8a. However, the temporal relations are important to depict an event in memory, and our context cluster is composed of an ordered sequence of instances, which represents a whole event with temporal relations,

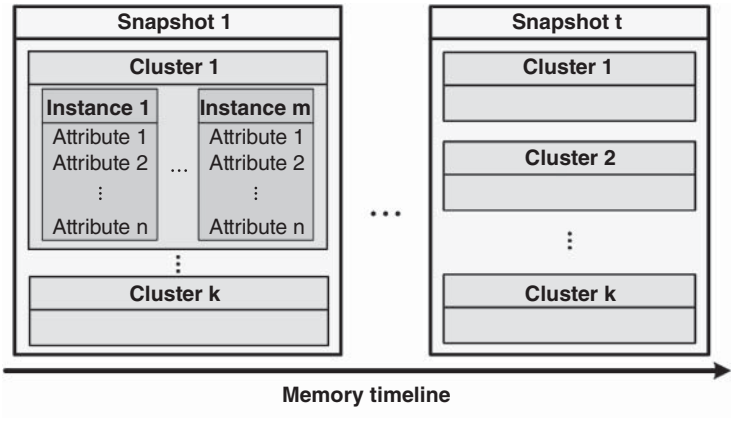


Figure 4.7 Context structure in evolving units of LCM. A snapshot is all the contents in memory at a specific time. The context snapshot evolves by time.

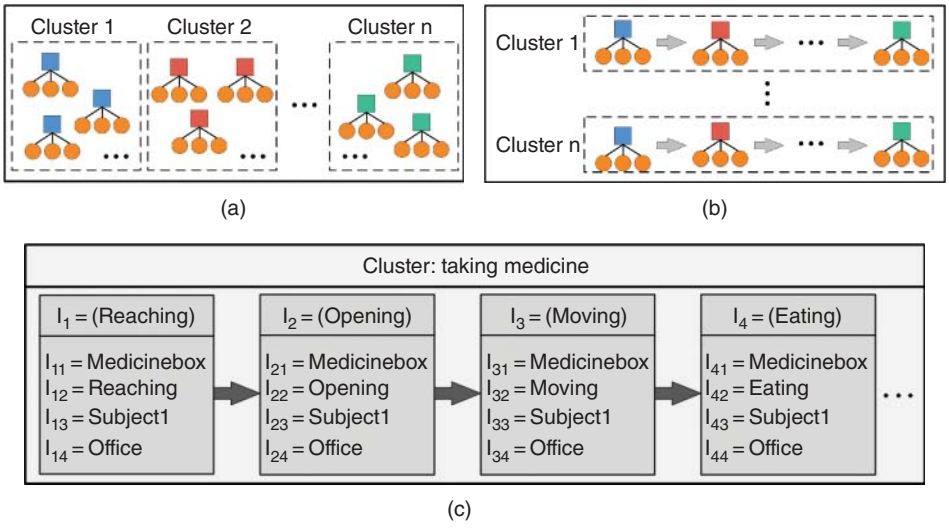


Figure 4.8 Different definitions of the *context cluster*. The similar instances are grouped into a cluster in CMM [37], as shown in (a). (b) shows the proposed cluster, which is an ordered sequence of instances, modeling the temporal relations. (c) shows an example of our cluster.

as shown in Figure 4.8b. One example is given in Figure 4.8c, where the activity of taking medicine is described by a series of consecutive semantic segments (instances). The organized structures and relations are then used for activity recognition.

4.3.4 Graph-based Event Video Summarization

4.3.4.1 Framework

Figure 4.9 gives an overview of our proposed mashup system [40]. Given a collection of videos, the shots of videos are first grouped into shot cliques using a graph-based near-duplicate detection approach. Then the semantics of those identified shot cliques

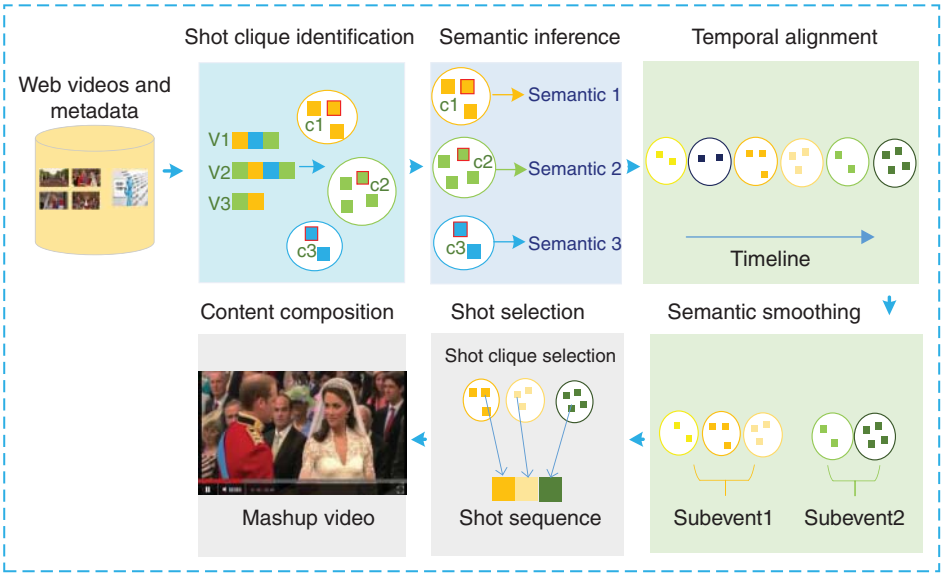


Figure 4.9 The flowchart of the proposed event video mashup approach.

are inferred to unveil their relevance to key semantic facets of the event. Next, the shot cliques are temporally aligned, followed by a semantic smoothing step to refine the alignment and discover key subevents of the event. Taking into consideration both importance and diversity factors, a shot clique selection method is performed. From each top ranked shot clique, a shot is chosen to represent the clique. All the above steps can be generalized as content selection. Finally, a sequence of selected shots in their previously determined order compose the final mashup video, with transition smoothness being considered. The mashup video length can be adaptively adjusted by varying the number of top ranked shot cliques. Among these components, graph-based temporal alignment is a key component. Next, we introduce it.

4.3.4.2 Temporal Alignment

In this subsection, we aim to align the shot cliques in a temporal order.

First, we build a matrix $L \in \mathbb{R}^{n\times n}$ based on the pairwise temporal orders of shot cliques obtained from the original videos as:

$$L_{i,j} = \begin{cases} 1 & \text{if } s_i \text{ is before } s_j, \\ -1 & \text{if } s_i \text{ is after } s_j, \\ 0 & \text{if not determined} \end{cases} \tag{4.34}$$

where $L_{i,j}$ is element (i, j) of L, s_i denotes the ith shot clique, and there are n shot cliques. The temporal orders of some shot clique pairs cannot be directly observed from the original videos, but they may be inferred via some bridging shot cliques. Based on this, we conduct matrix completion for L:

$$L_{i,j} = \begin{cases} 0 \to +1 & \text{if } L_{i,p} = 1, \ L_{p,j} = 1 \\ 0 \to -1 & \text{if } L_{i,p} = -1, \ L_{p,j} = -1 \end{cases} \tag{4.35}$$

Next, we can align the shot cliques using the local temporal information contained in L. Specifically, we assign a temporal position t_i ($t_i \in \{1, 2, \cdots, n\}, t_i \neq t_j \ \ if \ \ i \neq j$) to each shot clique s_i in order to maximize the match between the mined temporal orders L and the estimated temporal orders:

$$t = \underset{t_i, t_j \in \{1,2,\ldots,n\}}{\arg\max} \sum_{i=1}^{n} \sum_{j=1}^{n} sgn(t_j - t_i) L_{i,j} \tag{4.36}$$

where $\text{sgn}(x)$ is a sign function:

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{if } x < 0 \end{cases} \tag{4.37}$$

Eq. (4.36) is a non-deterministic polynomial-time hardness (NP-hard) problem. Alternatively, we propose an algorithm to approximate the optimal solution. To be specific, we first initialize the temporal positions of the shot cliques randomly. And then swap the temporal positions t_i and t_j of s_i and s_j iteratively as long as (i) t_i and t_j contradict with $L_{i,j}$ and (ii) the swap can increase the objective function.

We regard all the shot cliques as a graph, where the vertices are shot cliques and an edge is added between two vertices s_i and s_j, if $L_{i,j} \neq 0$. Then the graph can be divided into a set of connected subgraphs, which means the inter-temporal-orders of the subgraphs cannot be determined from the videos. In fact, these subgraphs often correspond to independent components of the events. Since users' interest and focus towards an event change with the evolution of the event, we can employ users' interest trend to estimate the temporal orders of the independent sets of shot cliques. Upload time of video is a good indicator of users' interest trend about an event. Therefore, we calculate the average upload time of the videos related to the subgraphs to determine the temporal orders.

4.3.5 TGIF: A New Dataset and Benchmark on Animated GIF Description

In the previous subsections, we described our methods for video understanding, while in this subsection, we introduce a dataset collected by ourselves. With the recent popularity of animated GIFs on social media, there is need for ways to index them with rich metadata. To advance research on animated GIF understanding, we collected a new dataset, Tumblr GIF (TGIF) [41], with 100K animated GIFs from Tumblr and 120K natural language descriptions obtained via crowdsourcing. The motivation for this work is to develop a testbed for image sequence description systems, where the task is to generate natural language descriptions for animated GIFs or video clips.

4.3.5.1 Data Collection

We extracted a year's worth of GIF posts from Tumblr using the public API,[3] and cleaned up the data with four filters:

- **Cartoon.** Filters out cartoon content by matching popular animation keywords to user tags.
- **Static.** Discards GIFs that show little to no motion (basically static images). To detect static GIFs, we manually annotated 7K GIFs as either static or dynamic, and trained a random forest classifier based on C3D features [42]. The 5-fold cross validation accuracy for this classifier is 89.4%.

3 https://www.tumblr.com/docs/en/api/v2

- **Text.** Filters out GIFs that contain text, e.g., memes, by detecting text regions using the extremal regions detector [43] and discarding a GIF if the regions cover more than 2% of the image area.
- **Dedup.** Computes 64bit discrete cosine transform (DCT) image hash using pHash [44] and applys multiple index hashing [45] to perform a k-nearest-neighbor search ($k = 100$) in the Hamming space. A GIF is considered a duplicate if there are more than 10 overlapping frames with other GIFs. On a held-out dataset, the false alarm rate is around 2%.

Finally, we manually validated the resulting GIFs to see whether there was any cartoon, static, or textual content. Each GIF was reviewed by at least two annotators. After these steps, we obtained a corpus of 100K clean animated GIFs.

4.3.5.2 Data Annotation

We annotated animated GIFs with natural language descriptions using the crowdsourcing service CrowdFlower. We carefully designed our annotation task with various quality control mechanisms to ensure the sentences were both syntactically and semantically of high quality.

A total of 931 workers participated in our annotation task. We only allowed workers from Australia, Canada, New Zealand, UK and the USA in an effort to collect fluent descriptions from native English speakers. Figure 4.10 shows the instructions given to

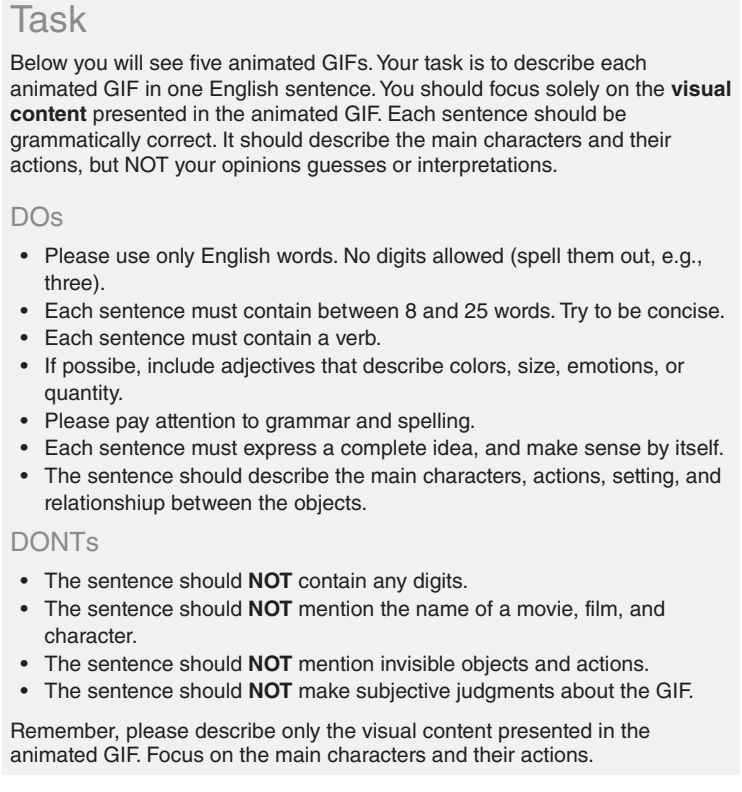


Figure 4.10 The instructions for the crowdworkers.

the workers. Each task showed five animated GIFs and asked the worker to describe each with one sentence. To promote language style diversity, each worker could rate no more than 800 images (0.7% of our corpus). We paid 0.02 USD per sentence; the entire crowdsourcing cost less than 4K USD.

Syntactic validation Since the workers provided free-form text, we automatically validated the sentences before submission. We checked that each sentence

- contains at least 8, but no more than 25 words (white space separated)
- contains only ASCII characters
- does not contain profanity (checked by keyword matching)
- is typed, not copy/pasted (checked by disabling copy/paste on the task page)
- contains a main verb (checked by using standard position tagging [46])
- contains no named entities, such as the name of an actor/actress, movie, or country (checked by the Named Entity Recognition results from DBpedia spotlight [47]) and
- is grammatical and free of typographical errors (checked by the LanguageTool[4]).

This validation pipeline ensures sentences are *syntactically* good. But it does not ensure their *semantic* correctness, i.e., there is no guarantee that a sentence accurately describes the corresponding GIF. We therefore designed a semantic validation pipeline, described next.

Semantic validation Ideally, we wanted to validate the semantic correctness of every submitted sentence (as we did for syntactic validation), but this was impractical. Instead we used the "blacklisting" approach, where we identified workers who underperformed and blocked them accordingly.

We annotated a small number of GIFs and used them to measure the performance of workers. We collected a validation dataset with 100 GIFs and annotated each with 10 sentences using CrowdFlower. We carefully hand-picked the GIFs whose visual story was clear and unambiguous. After collecting the sentences, we manually reviewed and edited them to make sure they met our standard.

Using the validation dataset, we measured the semantic relatedness of sentence to GIF using METEOR [48], a metric commonly used within the natural language processing (NLP) community to measure machine translation quality. We compare a user-provided sentence to 10 reference sentences using the metric and accepted a sentence if the METEOR score was above a threshold (empirically set at 20%). This filtered out junk sentences, e.g., "this is a funny GIF taken in a nice day," but retained sentences with similar semantic meaning as the reference sentences.

We used the dataset in both the qualification and the main tasks. In the qualification task, we provided five GIFs from the validation dataset and approved a worker if they successfully described at least four tests. In the main task, we randomly mixed one validation question with four main questions; a worker was blacklisted if the overall approval rate on validation questions fell below 80%. Because validation questions were indistinguishable from normal task questions, workers had to continue to maintain a high level of accuracy in order to remain eligible for the task.

As we ran the CrowdFlower task, we regularly reviewed failed sentences and, in the case of a false alarm, we manually added the failed sentence to the reference

4 https://languagetool.org/

sentence pool and removed the worker from the blacklist. Rashtchian et al. [49] and Chen et al. [50] used a similar prescreening strategy to approve crowdworkers; our strategy to validate sentences *during* the main task is unique to our work.

4.3.6 Experiments

We evaluated our algorithm on the task of image and video annotation. Due to the space limitations, we do not show the results for sections 4.3.3 and 4.3.4. First, we studied the influence of the parameters in our algorithm. Then, we compared our results with state-of-the-art algorithms on four standard datasets.

4.3.6.1 Experimental Settings

4.3.6.1.1 Datasets

We considered four publicly available datasets that have been widely used in previous work.

IXMAS This video dataset consists of 12 action classes (e.g., check watch, cross arms, and scratch head). Each action is executed three times by 12 actors and is recorded with five cameras observing the subjects from very different perspectives.

NUS-WIDE This image dataset contains 269,648 images downloaded from Flickr. Tagging ground truth for 81 semantic concepts is provided for evaluation. Similar to [51], we only use the images associated with the 10 most frequent concept annotations, obtaining 167,577 images in total.

For IXMAS, we use half of the tagged data to infer the tags for the rest of the data. For the other dataset, out-of-sample extension is utilized to annotate the test data points. We randomly selected 10,000 data points for training, and used the rest for testing. In the training dataset, partial data points are with tags.

4.3.6.1.2 Features

For IXMAS, we extracted the 500-D bag of words (BoW) feature based on SIFT for each camera, and the five cameras are taken as five features, following [52]. For NUS-WIDE, six types of low-level features are provided, including 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments, and 500-D BoW based on SIFT descriptions.

4.3.6.1.3 Baseline Methods and Evaluation Metrics

To evaluate the performance of the optimal graph, we compared OGL with different graph construction methods, e.g., LGC [31], LLE [53], and L2graph [54]. We further extended and compared all these methods on larger datasets by out-of-sample extension. We also compared them against TagProp [55] and fastTag [56] algorithms, which currently achieve the best performance on the benchmark datasets. To test different fitting models, we combined OGL with fastTag to check the performance. We also compared them with MBRM [57] and JEC [58].

We evaluated our models with standard performance measures:

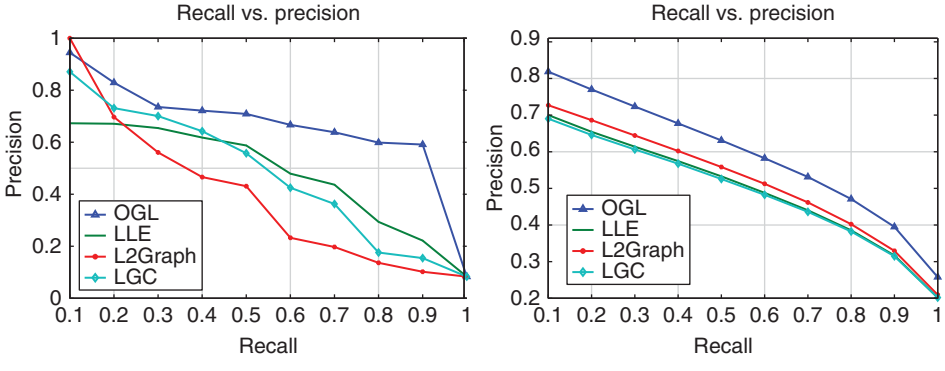
Precision vs recall over each label is calculated by computing the precision at each recall.

Mean average precision (MAP) over labels is calculated by computing for each label the average of the precisions measured after each relevant image is retrieved.

4.3.6.2 Results

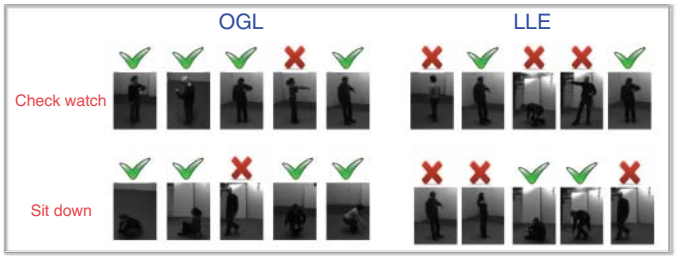
Figure 4.11 shows the comparisons for the two datasets. We also show some qualitative results in Figure 4.11c. From these figures, we have the following observations:

- By learning an optimal graph from partial tags and multiple features, the performance for image and video annotation is improved significantly compared with other graph construction methods. This indicates that using partial tags and multiple features can result in a better graph.
- The improvement of OGL over other methods is more significant for the IXMAS dataset than for the other datasets. This is probably because out-of-sample extension will reduce the performance gaps of different methods. L2Graph is a strong competitor in all cases, while LGC achieves the worst performance in all datasets. This indicates that a reconstruction coefficients-based graph is better than pairwise distance-based graphs in these datasets.
- Compared with other state-of-the-art non-graph-based methods, our OGL outperforms most of the existing approaches and achieves comparable performance with the current best result.



(a) Results on IXMAS

(b) Results on NUS-WIDE



(c) Qualitative results on IXMAS dataset

Figure 4.11 Experiment results on four datasets.

- From the qualitative results in Figure 4.11c, we can see that OGL can annotate the videos more precisely in the IXMAS dataset than LLE. This is because OGL makes use of both tags and multiple features, and automatically decides their weights, while LLE considers visual features only. In this case, OGL is more robust to some inaccurate visual features (e.g., light) or the tags, but LLE is more sensitive to these changes.

4.4 Conclusions and Future Work

In this work we cover video annotation, retrieval, and summarization, three approaches to fight the consequences of big video data. We present the state-of-the-art research in each of the research field. Specifically, we discuss both unsupervised and semi-supervised methods to facilitate video understanding tasks. We consider two general research problems: video retrieval and video annotation/summarization. While the former focuses on designing efficient algorithms for retrieving useful information from large-scale data, the latter aims to reduce the size of the data. Video retrieval provides new models and methods for effectively and efficiently searching through the huge variety of video data that are available in different kinds of repositories (digital libraries, Web portals, social networks, multimedia databases, etc.). On the other hand, video annotation/summarization generates annotations or a short summary of a video. It provides efficient storage and quick browsing of large collection of video data without losing important aspects. Depending on whether or not human labels are involved, video retrieval and annotation/summarization can be further categorized into unsupervised and semi-supervised based methods. In the future, we will consider applying our video understanding methods to more real-world applications, e.g., surveillance, video advertising, and short video recommendations.

References

1 Song, J., Yang, Y., Yang, Y., Huang, Z., and Shen, H.T. (2013) Inter-media hashing for large-scale retrieval from heterogeneous data sources, in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*, New York, USA, pp. 78–796, ACM, New York.

2 Song, J., Yang, Y., Li, X., Huang, Z., and Yang, Y. (2014) Robust hashing with local models for approximate similarity search. *IEEE Transactions on Cybernetics*, 44 (7), 1225–1236.

3 Wang, J., Zhang, T., Song, J., Sebe, N., and Shen, H.T. (2017) A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (4), 769–790.

4 Song, J., Gao, L., Liu, L., Zhu, X., and Sebe, N. (2017) Quantization-based hashing: a general framework for scalable image and video retrieval. *Pattern Recognition*, 75, 175–187.

5 Sidiropoulos, P., Vrochidis, S., and Kompatsiaris, I. (2011) Content-based binary image retrieval using the adaptive hierarchical density histogram. *Pattern Recognition*, 44 (4), 739–750.

6 Ye, G., Liu, D., Wang, J., and Chang, S.F. (2013) Large-scale video hashing via structure learning, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2272–2279.

7 Song, J., Yang, Y., Huang, Z., Shen, H., and Hong, R. (2011) Multiple feature hashing for real-time large scale near-duplicate video retrieval, in *ACM Multimedia*, ACM, pp. 423–432.

8 Cao, L., Li, Z., Mu, Y., and Chang, S.F. (2012) Submodular video hashing: a unified framework towards video pooling and indexing, in *Proceedings of the 20th ACM International Conference on Multimedia (MM '12), Nara, Japan*, pp. 299–308, ACM, New York

9 Girshick, R. (2015) Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448.

10 You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016) Image captioning with semantic attention, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4651–4659.

11 Liong, V.E., Lu, J., Tan, Y.P., and Zhou, J. (2016) Deep video hashing. *IEEE Transactions on Multimedia*, 19 (6), 1209–1219.

12 Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015) Sequence to sequence-video to text, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4534–4542.

13 Pan, P., Xu, Z., Yang, Y., Wu, F., and Zhuang, Y. (2016) Hierarchical recurrent neural encoder for video representation with application to captioning, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1029–1038.

14 Gu, Y., Ma, C., and Yang, J. (2016) Supervised recurrent hashing for large scale video retrieval, in *Proceedings of the 2016 ACM on Multimedia Conference*, ACM, pp. 272–276.

15 Zhang, H., Wang, M., Hong, R., and Chua, T.S. (2016) Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing, in *Proceedings of the 2016 ACM on Multimedia Conference*, ACM, pp. 781–790.

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

17 Song, J., Yang, Y., Huang, Z., Shen, H.T., and Luo, J. (2013) Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia*, 15 (8), 1997–2008.

18 Wu, X., Hauptmann, A.G., and Ngo, C.W. (2007) Practical elimination of near-duplicates from web video search, in *Proceedings of the 15th ACM International Conference on Multimedia (MM '07), Augsburg, Germany*, pp. 218–227, ACM, New York.

19 Boureau, Y., Roux, N.L., Bach, F.R., Ponce, J., and LeCun, Y. (2011) Ask the locals: Multi-way local pooling for image recognition, in *Proceedings of the 2011 International Conference on Computer Vision (ICCV '11)*, pp. 2651–2658, IEEE Computer Society, Washington, DC.

20 Zhu, X., Ghahramani, Z., and Lafferty, J.D. (2003) Semi-supervised learning using Gaussian fields and harmonic functions, in *Proceedings of the 20th International Conference on International Conference on Machine Learning (ICML'03)*, Washington, DC, USA, pp. 912–919, AAAI Press.

21 Cc_web_video: Near-duplicate web video dataset, http://vireo.cs.cityu.edu.hk/webvideo.

22 Shang, L., Yang, L., Wang, F., Chan, K.P., and Hua, X.S. (2010) Real-time large scale near-duplicate web video retrieval, in *Proceedings of the 18th ACM International Conference on Multimedia (MM '10)*, Florence, Italy, pp. 531–540, ACM, New York.

23 Zhang, D., Wang, J., Cai, D., and Lu, J. (2010) Self-taught hashing for fast similarity search, in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, Geneva, Switzerland, pp. 18–25, ACM, New York.

24 Weiss, Y., Torralba, A., and Fergus, R. (2008) *Spectral hashing*, in *Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS'08)*, Vancouver, British Columbia, Canada, pp. 1753–1760, Curran Associates Inc.

25 Jiang, Y.G., Ye, G., Chang, S.F., Ellis, D., and Loui, A.C. (2011) Consumer video understanding: A benchmark database and an evaluation of human and machine performance, in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval (ICMR '11)*, Trento, Italy, pp. 29:1–29:8, ACM, New York.

26 Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016) Youtube-8m: A large-scale video classification benchmark. *Computing Research Repository (CoRR), abs/1609*.08675.

27 Yang, Y., Wang, Z., Yang, J., Wang, J., Chang, S., and Huang, T.S. (2014) Data clustering by Laplacian regularized l1-graph, in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*, Quebec, Canada, pp. 3148–3149, AAAI Press.

28 Cheng, B., Yang, J., Yan, S., Fu, Y., and Huang, T.S. (2010) Learning with l1-graph for image analysis. *IEEE Transactions on Image Processing*, 19 (4), 858–866.

29 Deng, C., Ji, R., Tao, D., Gao, X., and Li, X. (2014) Weakly supervised multi-graph learning for robust image reranking. *IEEE Transactions on Multimedia*, 16 (3), 785–795.

30 Liu, W., Wang, J., and Chang, S.F. (2012) Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100 (9), 2624–2638.

31 Zhou, D., Bousquet, O., Lal, T.N., Weston, J., and Schölkopf, B. (2003) Learning with local and global consistency, in *Proceedings of the 16th International Conference on Neural Information Processing Systems, (NIPS '03)*, Whistler, British Columbia, Canada, pp. 321–328, MIT Press, Cambridge, MA, in Proceedings of the 16th International Conference on Neural Information *Processing Systems, (NIPS'03)*, Whistler, British Columbia, Canada, pp. 321–328, MIT Press, Cambridge, MA.

32 Belkin, M., Niyogi, P., and Sindhwani, V. (2006) Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7, 2399–2434.

33 Nie, F., Xu, D., Tsang, I.W.H., and Zhang, C. (2010) Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction. *TIP*, 19 (7), 1921–1932.

34 Tian, Q. and Chen, S. (2017) Cross-heterogeneous-database age estimation through correlation representation learning. *Neurocomputing*, 238, 286–295.

35 Pan, Z., Jin, P., Lei, J., Zhang, Y., Sun, X., and Kwong, S. (2016) Fast reference frame selection based on content similarity for low complexity HEVC encoder. *Visual Communication and Image Representation* 40, 516–524.

36 Song, J., Gao, L., Nie, F., Shen, H.T., Yan, Y., and Sebe, N. (2016) Optimized graph learning using partial tags and multiple features for image and video annotation. *Transactions Image Processing*, 25 (11), 4999–5011

37 Deng, T., Zhao, L., Wang, H., Liu, Q., and Feng, L. (2013) Refinder: A context-based information refinding system. *IEEE Transactions on Knowledge and Data Engineering*, 25 (9), 2119–2132, doi: 10.1109/TKDE.2012.157.

38 Wang, L., Zhao, X., Si, Y., Cao, L., and Liu, Y. (2017) Context-associative hierarchical memory model for human activity recognition and prediction. *IEEE Transactions on Multimedia*, 19 (3), 646–659.

39 Sperling, G. (1963) A model for visual memory tasks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 5 (1), 19–31.

40 Gao, L., Wang, P., Song, J., Huang, Z., Shao, J., and Shen, H.T. (2017) Event video mashup: From hundreds of videos to minutes of skeleton, in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, February 4–9, San Francisco, California, USA, pp. 1323–1330.

41 Li, Y., Song, Y., Cao, L., Tetreault, J.R., Goldberg, L., Jaimes, A., and Luo, J. (2016) TGIF: A new dataset and benchmark on animated GIF description, in in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 27–30, Las Vegas, NV, USA, pp. 4641–4650.

42 Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., and Paluri, M. (2015) Learning spatiotemporal features with 3D convolutional networks, in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV '15)*, pp. 4489–4497, EEE Computer Society, Washington, DC.

43 Neumann, L. and Matas, J. (2012) Real-time scene text localization and recognition, in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3538–3545, IEEE Computer Society, Washington, DC.

44 Zauner, C. (2010) *Implementation and benchmarking of perceptual image hash functions*, Upper Austria University of Applied Sciences, Hagenberg Campus.

45 Norouzi, M., Punjani, A., and Fleet, D.J. (2014) Fast exact search in hamming space with multi-index hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (6), 1107–1119.

46 Toutanova, K. and Manning, C.D. (2010) Enriching the knowledge sources used in a maximum entropy part-of-speech tagger, in *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics (EMNLP '13), Volume 13*, Hong Kong, pp. 63–70, Association for Computational Linguistics, Stroudsburg, PA.

47 Daiber, J., Jakob, M., Hokamp, C., and Mendes, P.N. (2013) Improving efficiency and accuracy in multilingual entity extraction, in *Proceedings of the 9th International Conference on Semantic Systems (I-SEMANTICS '13), Graz, Austria*, pp. 121–124, ACM, New York.

48 Lavie, M.D.A. (2014) Meteor universal: Language specific translation evaluation for any target language, in *Proceedings of the Ninth Workshop on Statistical Machine Translation, Association for Computational Linguistics*, Baltimore, Maryland, USA, pp. 376–380,

49 Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. (2010) Collecting image annotations using Amazon's mechanical turk, in *Proceedings of the NAACL HLT*

2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (CSLDAMT '10), Los Angeles, California, pp. 139–147, Association for Computational Linguistics, Stroudsburg, PA.

50 Chen, D.L. and Dolan, W.B. (2011) Collecting highly parallel data for paraphrase evaluation, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11)*, Volume 1, Portland, Oregon, pp. 190–200, Association for Computational Linguistics, Stroudsburg, PA.

51 Liu, W., Wang, J., Kumar, S., and Chang, S. (2011) Hashing with graphs, in *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML '11)*, Bellevue, Washington, USA, pp. 1–8, Omnipress.

52 Yan, Y., Ricci, E., Subramanian, R., Liu, G., and Sebe, N. (2014) Multitask linear discriminant analysis for view invariant action recognition. *IEEE Transactions on Image Processing*, 23 (12), 5599–5611.

53 Roweis, S. and Saul, L. (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500), 2323–2326.

54 Peng, X., Zhang, L., and Yi, Z. (2012) Constructing l2-graph for subspace learning and segmentation. *Computing Research Repository*, abs/1209.0841.

55 Guillaumin, M., Mensink, T., Verbeek, J.J., and Schmid, C. (2009) Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in *2009 IEEE 12th International Conference on Computer Vision*, pp. 309–316, IEEE.

56 Chen, M., Zheng, A., and Weinberger, K.Q. (2013) Fast image tagging, in *Proceedings of Machine Learning Research*, 28 (3), 1274–1282.

57 Feng, S., Manmatha, R., and Lavrenko, V. (2004) Multiple Bernoulli relevance models for image and video annotation., in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Volume* 2, pp. II–II, IEEE.

58 Makadia, A., Pavlovic, V., and Kumar, S. (2010) Baselines for image annotation. *International Journal of Computer Vision*, 90 (1), 88–105.