Fast Action Retrieval from Videos via Feature Disaggregation

Jie Qin[1]
qinjiebuaa@gmail.com
Li Liu[2]
li2.liu@northumbria.ac.uk
Mengyang Yu[2]
m.y.yu@ieee.org
Yunhong Wang[1]
yhwang@buaa.edu.cn
Ling Shao[2]
ling.shao@ieee.org

[1] Beijing Key Laboratory of Digital Media, School of Computer Science and Engineering, Beihang University, China

[2] Computer Vision and Artificial Intelligence Group, Department of Computer Science and Digital Technologies, Northumbria University, UK

Abstract

Learning based hashing methods, which aim at learning similarity-preserving binary codes for efficient nearest neighbor search, have been actively studied recently. A majority of the approaches address hashing problems for image collections. However, due to the extra temporal information, videos are usually represented by much higher dimensional (thousands or even more) features compared with images, causing high computational complexity for conventional hashing schemes.

In this paper, we propose a simple and efficient hashing scheme for high-dimensional video data. This method, called Disaggregation Hashing, exploits the correlations among different feature dimensions. An intuitive feature disaggregation method is first proposed, followed by a novel hashing algorithm based on different feature clusters. We demonstrate the efficiency and effectiveness of our method by theoretical analysis and exploring its application on action retrieval from video databases. Extensive experiments show the superiority of our binary coding scheme over state-of-the-art hashing methods.

1 Introduction

With the rapid development of digital media, massive collections of images/videos are created every day. This poses lots of challenging problems to the computer vision community, among which one major challenge is, given an image/video, how to efficiently find its similar ones (*i.e.*, similarity search). One conventional way to address the problem is utilizing nearest neighbour search. Tree-based schemes have also been widely studied to improve the efficiency of searching. However, these techniques are not scalable to high-dimensional data as they cannot easily find the approximate structure of the sparse data in the high-dimensional space. To overcome this challenge, a variety of hashing algorithms have been actively studied. The hashing techniques aim at efficiently embedding data from the high-dimensional

space to a low-dimensional space, while preserving similarities among data points. After obtaining compact data representations (*i.e*., binary codes), approximate nearest neighbor (ANN) search can operate with constant or sub-linear time complexity.

Locality Sensitive Hashing (LSH) [4] is one of the most widely employed hashing methods. LSH is based on random projections that project data points that are close in the Euclidean space to similar codes. However, its performance is limited because the data distribution is not taken into account. Thus, a number of data-dependent hashing methods have been proposed, aiming at better fitting data distributions to achieve better performance. For instance, Spectral Hashing [20] proposed by Weiss *et al*. formalizes the hashing task as a particular form of graph partitioning, in which the Laplace-Beltrami eigenfunctions of manifolds are used to determine binary codes. Motivated by Weiss *et al*., Liu *et al*. [11] utilize Anchor Graphs to learn compact binary codes by directly approximating the sparse neighborhood graph and its associated adjacency matrix. Another popular hashing approach named iterative quantization (ITQ) [2] has connections to the Procrustes problem and performs compact binary embedding by minimizing the quantization error of rotating zero-centered data projected by principal component analysis (PCA). Apart from the above approaches which are based on hyperplanes, Heo *et al*. [3] introduce a hashing scheme based on hyperspheres to map more spatially coherent data points into a binary code.

However, a majority of hashing techniques have been specifically developed for image retrieval. Compared with images, representations of videos can be more sophisticated and complex. By employing leading feature encoding techniques (*e.g*., Bag of Words (BoW) [8], Vector of Locally Aggregated Descriptors (VLAD) [6], and Fisher Vectors (FV) [14, 15]), representations of thousands or even more dimensions will be generated. Although the time complexity of ANN search can be significantly reduced by employing hashing techniques, learning hash functions in the training stage can become quite time-consuming and even intractable when dealing with very high-dimensional video data, which usually involves complex iterative optimization, eigen-decomposition, *etc*. Besides, a majority of hashing methods are based on linear projections (*e.g*., random projection [4]). In terms of mapping the data from the very high-dimensional space to a reduced one, the memory requirements for storing the projection matrix and performing the mapping operation impose heavy burdens. Taking the state-of-the-art hashing method, ITQ [2], as an example, to reduce 100,000-dimensional features to 10,000-dimensional ones, the memory cost for the projection matrix is about 7.5GB and the number of multiplications for coding a test data point is 10^9.

To overcome these shortcomings, in this paper, a novel hashing scheme, namely Disaggregation Hashing (DH), is proposed for high-dimensional video data by exploiting the correlations among different feature dimensions. We start with disaggregating the original high dimension into several low-dimensional feature clusters/groups where feature values are similar among all training data points. Subsequently, simple and efficient hash functions are learned to embed original feature vectors to compact binary codes separately based on each feature group. Since each group can be embedded independently, a substantial speed up can be guaranteed for learning hash functions in the training stage. Furthermore, in the testing stage, the proposed method only needs to store a D-dimensional projection vector and computing binary codes is of $O(D)$ complexity, where D is the dimension of the original data space. The main contributions of this paper include: (1) We propose a novel hashing scheme for high-dimensional video data, which outperforms state-of-the-art methods; (2) Feature disaggregation/clustering is proposed to group similar feature dimensions of high-dimensional data by exploiting correlations along the dimensions; (3) Hash functions, aiming at preserving the intrinsic data similarity, are learned independently using greedy optimiza-

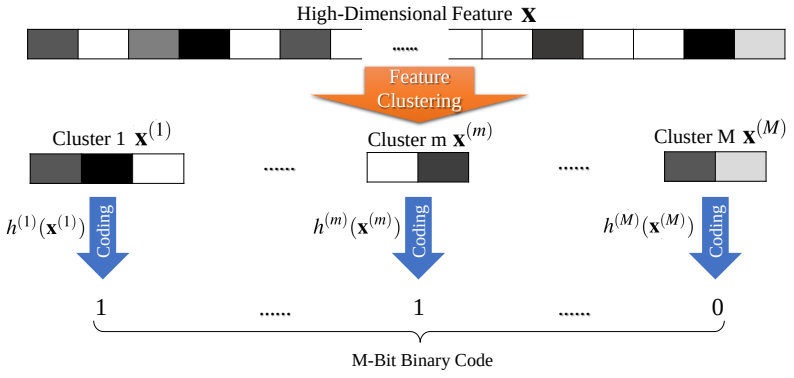


Figure 1: The overall framework of our hashing scheme (see text for more details).

tion on each feature group, which can significantly reduce the computational complexity and memory usage in both the training and testing stages.

2 Disaggregation Hashing

Given a set of N data points $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $\forall \mathbf{x}_n \in \mathbb{R}^D$ where $n = 1, ..., N$, our goal is to find a binary embedding function $H(\mathbf{x}) = \{-1, +1\}^M$, where M indicates the length of the code. Instead of learning and applying H on the entire D dimensions, our Disaggregation Hashing utilizes different hash functions on different groups of feature dimensions by exploiting the correlations among them. Thus, each bit of the entire code is learned independently, which enhances the scalability on high-dimensional video data and allows fast parallel computation as well. Figure 1 shows the overall framework of our method. In the following, feature clustering is first introduced and a novel hashing algorithm is proposed subsequently.

2.1 Clustering of Feature Dimensions

In terms of high-dimensional data, dimensionality reduction is often employed to avoid high time and space complexity. One of the most widely used unsupervised dimensionality reduction methods is PCA. By incorporating a special structure constraint into its formulation, the feature clustering problem (*i.e.*, merging similar feature dimensions into clusters as shown in Figure 1) can be solved by k-means clustering [1, 9, 21]. Subsequently, we briefly introduce the feature clustering strategy and show how it benefits the following hashing scheme.

Recall the formulation of PCA:

$$\max_{\mathbf{W}} \ \text{trace}(\mathbf{W}\mathbf{X}'\mathbf{X}'^T\mathbf{W}^T) \ s.t. \ \mathbf{W}\mathbf{W}^T = \mathbf{I} \tag{1}$$

where $\mathbf{X}' \in \mathbb{R}^{D\times N}$ represents the centralized $\mathbf{X}$ and $\mathbf{W} \in \mathbb{R}^{d\times D}$ is the projection matrix.

By reforming the structure of $\mathbf{W}$ that $w_{i,j} \in \{0, 1\}$ and $||\mathbf{w}_{(:,j)}||_0 = 1$ ($w_{i,j}$ and $\mathbf{w}_{(:,j)}$ are the (i, j) entry and j^{th} column of $\mathbf{W}$, respectively, and $||.||_0$ denotes the number of nonzero entries), $\mathbf{W}$ can be regarded as the operator for feature clustering. If $\mathbf{X}'$ is multiplied by $\mathbf{W}$, its feature dimensions are clustered into groups and one feature dimension only belongs to one group. Let $\mathbf{X}''$ be the matrix after performing feature clustering on $\mathbf{X}'$, then its entries

$x''_{i,j} = \sum_{\{k|w_{i,k}=1\}} x'_{k,j}$. However, directly incorporating this constraint into Eq. (1) will result in a mixed-integer problem ($\mathbf{W}\mathbf{W}^T \neq \mathbf{I}$). To make it more feasible to solve, a normalization factor $\delta_i = ||\mathbf{w}_{(i,:)}||_0$ for each row of $\mathbf{W}$ is employed. Thus, the structure constraint on $\mathbf{W}$ will become:

$$\mathbf{W} \in \{\mathbf{W}|w_{i,j} \in \{0, \frac{1}{\sqrt{\delta_i}}\} \forall i,\ \mathbf{W}\mathbf{W}^T = \mathbf{I}\} \tag{2}$$

After incorporating the above constraint into Eq. (1), we can find a solution to feature clustering that is identical to the one to PCA. Furthermore, as [1, 21] asserted, k-means clustering has the following formulation if we treat $\mathbf{W}$ as the cluster indicator:

$$\max_{\mathbf{W}} \text{trace}(\mathbf{W}\mathbf{Y}^T\mathbf{Y}\mathbf{W}^T) \tag{3}$$

Straightforwardly, we can simply regard $\mathbf{X}'^T \in \mathbb{R}^{N\times D}$ as $\mathbf{Y}$ in Eq. (3). Thereby, the feature clustering problem, *i.e.*, Eq. (1) with the substituted constraint in Eq. (2), can be effectively addressed by k-means clustering. In other words, by applying k-means clustering on D N-dimensional data points $\mathbf{X}'^T$, we can obtain the feature clustering indexes. Note that, if N is extremely large, applying k-means on such an $N \times D$ matrix is still difficult and time-consuming. We will address this problem in Section 2.2, along with the detailed hashing algorithm.

It is noteworthy that dimensions with similar feature values among all training data points tend to be clustered into the same group (*i.e.*, feature cluster) by applying the proposed feature clustering algorithm. Thus, correlations among different feature dimensions are automatically discovered and feature dimensions clustered in one group may have redundant information. Subsequently, a novel hashing scheme, namely Disaggregation Hashing, is proposed to remove the redundant information by embedding each group of features into a single binary code. Once the clustering indexes are obtained, each bit of the final binary code can be learned based on each group independently. Furthermore, with the help of parallel computing, the hashing scheme can be scalable on very high-dimensional data with a significant reduction of memory usage and computational complexity.

2.2 Hash Function Learning

After accomplishing feature clustering, the original data matrix $\mathbf{X}$ is split into M groups: $\mathbf{X}^{(m)} \in \mathbb{R}^{d_m\times N}$, where $m = 1,...,M$, and $\sum_{m=1}^{M} d_m = D$. In other words, the original feature space $\mathbb{R}^D$ is split into M subspaces $\mathbb{R}^{d_m}$. As aforementioned, each bit of the entire binary code is learned in terms of each feature group independently. Hence, the overall hash function $H(\mathbf{x})$ consists of M independent functions $(h^{(1)}(\mathbf{x}^{(1)}),...,h^{(M)}(\mathbf{x}^{(M)}))$, where $\mathbf{x}^{(m)}$ corresponds to the data point from the subspace $\mathbb{R}^{d_m}$. Each $h^{(m)}$ maps $\mathbf{x}^{(m)}$ into one binary code $c^{(m)} \in \{-1,+1\}$ and the final binary code is generated as a concatenation of M-bit codes $(c^{(1)},...,c^{(M)}) \in \{-1,+1\}^M$. Since each bit is learned independently, the requirement of bit uncorrelation [20] can be easily fulfilled. In the following, the formulation of $h^{(m)}(\mathbf{x}^{(m)})$ is introduced and we learn its parameters by addressing a greedy optimization problem.

Specifically, we define the above independent hash function as follows:

$$h^{(m)}(\mathbf{x}^{(m)}) = sgn(\mathbf{w}^{(m)}\mathbf{x}^{(m)} + b^{(m)}) \tag{4}$$

where the '$sgn()$' function returns '+1' if the argument is positive and '-1' otherwise, and $\mathbf{w}^{(m)} \in \mathbb{R}^{1\times d_m}$ is the d_m-dimensional projection vector.

Similar to the requirement of other hashing methods, a good code should have the property of similarity preserving, *i.e.*, mapping similar points in the original space to similar codes in the new space, which is commonly the Hamming space. Intuitively, our goal is to seek the codes that hold the minimal average Hamming distance between similar data points. Let $c_p^{(m)}$ denote one bit code for a data point $\mathbf{x}_p$, *i.e.*, $c_p^{(m)} = h^{(m)}(\mathbf{x}_p^{(m)}) = sgn(\mathbf{w}^{(m)}\mathbf{x}_p^{(m)} + b^{(m)})$, the aforementioned argument is equivalent to the following objective function:

$$\min \sum_{p,q} ||c_p^{(m)} c_q^{(m)} - l_{p,q}||^2 \tag{5}$$

where $p,q \in \{1,...,N\}$, and $c_p^{(m)}$, $c_q^{(m)}$ and $l_{p,q} \in \{-1,+1\}$. $l_{p,q} = +1$ if $\mathbf{x}_p$ and $\mathbf{x}_q$ are similar points (close to each other in the original space in the unsupervised setting) or come from the same semantic class (having the same class label in the semantic setting), and $l_{p,q} = -1$ otherwise.

Note that

$$\begin{aligned} \min \sum_{p,q} ||c_p^{(m)} c_q^{(m)} - l_{p,q}||^2 &= \min \sum_{p,q} (c_p^{(m)^2} c_q^{(m)^2} + l_{p,q}^2 - 2c_p^{(m)} c_q^{(m)} l_{p,q}) \\ &= \min \sum_{p,q} (1 + 1 - 2c_p^{(m)} c_q^{(m)} l_{p,q}). \end{aligned} \tag{6}$$

Therefore, Eq. (5) is equivalent to the following objective function:

$$\begin{aligned} \max \sum_{p,q} c_p^{(m)} c_q^{(m)} l_{p,q} &= \max_{\mathbf{w}^{(m)}, b^{(m)}} \sum_{p,q} sgn(\mathbf{w}^{(m)}\mathbf{x}_p^{(m)} + b^{(m)}) sgn(\mathbf{w}^{(m)}\mathbf{x}_q^{(m)} + b^{(m)}) l_{p,q} \\ &= \max_{\widehat{\mathbf{w}}^{(m)}} \sum_{p,q} sgn(\widehat{\mathbf{w}}^{(m)}\widehat{\mathbf{x}}_p^{(m)}) sgn(\widehat{\mathbf{w}}^{(m)}\widehat{\mathbf{x}}_q^{(m)}) l_{p,q} \end{aligned} \tag{7}$$

where $\widehat{\mathbf{w}}^{(m)} = [\mathbf{w}^{(m)}, b^{(m)}]$, $\widehat{\mathbf{x}}_p^{(m)} = [\mathbf{x}_p^{(m)^T}, 1]^T$ and $\widehat{\mathbf{x}}_q^{(m)} = [\mathbf{x}_q^{(m)^T}, 1]^T$.

Since the '$sgn()$' function is non-differentiable, Eq. (7) is still difficult to solve. Here, the spectral relaxation trick [12, 17] is applied and the '$sgn()$' function is approximated by using its signed magnitude, *i.e.*, $sgn(x) = x$. Therefore, by dropping the '$sgn()$' function, Eq. (7) can be relaxed as:

$$\max_{\widehat{\mathbf{w}}^{(m)}} \sum_{p,q} (\widehat{\mathbf{w}}^{(m)}\widehat{\mathbf{x}}_p^{(m)})(\widehat{\mathbf{w}}^{(m)}\widehat{\mathbf{x}}_q^{(m)}) l_{p,q} = \max_{\widehat{\mathbf{w}}^{(m)}} \widehat{\mathbf{w}}^{(m)} \widehat{\mathbf{X}}^{(m)} \mathbf{L} \widehat{\mathbf{X}}^{(m)^T} \widehat{\mathbf{w}}^{(m)^T} \tag{8}$$

where $\mathbf{L}$ is the similarity matrix with $l_{p,q}$ as its entries. Without loss of generality, we also assume $\|\widehat{\mathbf{w}}^{(m)}\| = 1, \forall m$.

It is noteworthy that Eq. (8) turns into a standard eigen-decomposition problem. Hence, the optimal solution for $\widehat{\mathbf{w}}^{(m)}$ can be obtained in a closed-form as the eigenvector of $\mathbf{Z}^{(m)} = \widehat{\mathbf{X}}^{(m)} \mathbf{L} \widehat{\mathbf{X}}^{(m)^T}$ associated with the largest eigenvalue. Although the optimization method is not a global one, it yields a practical approximation to the optimal solution to Eq. (8). Meanwhile, since $\mathbf{Z}^{(m)}$ is a $(d_m + 1) \times (d_m + 1)$ matrix and d_m is much smaller than D, computational expenses can be significantly reduced.

In the unsupervised setting, $\mathbf{L}$ cannot be explicitly acquired. So a pseudo $\mathbf{L}$ is employed based on the posteriori smoothness assumption, that is, the posterior probability $P(l|\mathbf{x})$ varies smoothly over the data manifold. Therefore, a sample's label l tends to be identical to its k

Algorithm 1: Disaggregation Hashing

Run k-means clustering on $\mathbf{X}$ to obtain the data clustering centroids matrix $\widehat{\mathbf{D}}$ and similarity matrix $\mathbf{L}$;
Run k-means clustering on $\widehat{\mathbf{D}'}^T$ to obtain the feature clustering indexes and split $\mathbf{X}$ into M groups: $\mathbf{X}^{(1)},...,\mathbf{X}^{(M)}$;
for $m = 1 : M$ **do**
    $\widehat{\mathbf{X}}^{(m)} \leftarrow [\mathbf{X}^{(m)^T}, 1^{N\times 1}]^T$;
    $\widehat{\mathbf{w}}^{(m)} \leftarrow$ The eigenvector of $\widehat{\mathbf{X}}^{(m)}\mathbf{L}\widehat{\mathbf{X}}^{(m)^T}$ associated with the largest eigenvalue;
    $[\mathbf{w}^{(m)}, b^{(m)}] \leftarrow \widehat{\mathbf{w}}^{(m)}$, $h^{(m)}(\mathbf{x}^{(m)}) = sgn(\mathbf{w}^{(m)}\mathbf{x}^{(m)} + b^{(m)})$;
end
return $H(\mathbf{x}) = (h^{(1)}(\mathbf{x}^{(1)}),...,h^{(M)}(\mathbf{x}^{(M)}))$.

nearest neighbors. In particular, k-means clustering is utilized on the original training set to find the centroids and $l_{p,q}$ equals '+1' if sample $\mathbf{x}_p$ and $\mathbf{x}_q$ belong to the same cluster.

Recall the feature clustering strategy introduced in Section 2.1, a problem still needs to be addressed, that is, k-means clustering on $\mathbf{X}'^T \in \mathbb{R}^{N\times D}$ can be of high computational complexity when N is very large. A small trick is applied here: assuming $\mathbf{d}_1,...,\mathbf{d}_K$ are the K centroid vectors after clustering the original training set $\mathbf{X}$ and these vectors are concatenated to form $\widehat{\mathbf{D}} \in \mathbb{R}^{D\times K}$, then k-means clustering is applied on $\widehat{\mathbf{D}'}^T$ instead of $\mathbf{X}'^T$, where $\widehat{\mathbf{D}'}$ is the centralized $\widehat{\mathbf{D}}$. The rationale behind this is that the K centroids can well represent the distribution of the original feature space. Therefore, applying feature clustering on the K centroids will lead to an approximate result, while reducing the computational loads significantly. Algorithm 1 illustrates the whole learning process of our method.

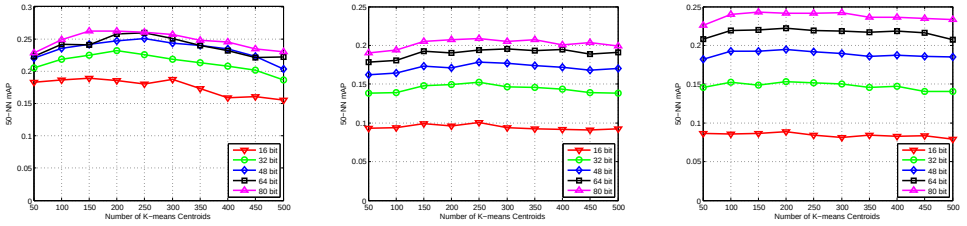
2.3 Complexity Analysis

The computational complexity of Disaggregation Hashing in the training stage mainly consists of two parts. The first part is for k-means clustering with P iterations on $\mathbf{X} \in \mathbb{R}^{D\times N}$ and $\widehat{\mathbf{D}'}^T \in \mathbb{R}^{K\times D}$, whose complexities are $O(PNKD)$ and $O(PDMK)$, where M is the code length. The second part is for learning hash functions on each group of features. This part consists of standard eigen-decompositions, having the complexity $O(\sum_{m=1}^{M} {d_m}^3)$, where $\sum_{m=1}^{M} d_m = D$. Note that hashing algorithms based on linear projections have at least $O(D^3)$ complexity and $O(\sum_{m=1}^{M} {d_m}^3) \ll O(D^3)$ when D is very large, which is very common in video retrieval applications. Hence, our hashing scheme is much more efficient than other methods in terms of learning hash functions. Besides, in the testing stage, given a query, our DH method only needs $O(\sum_{m=1}^{M} d_m) = O(D)$ complexity to obtain the binary codes.

3 Experiments and Results

3.1 Datasets and Experimental Protocol

The experiments are performed for the task of action retrieval from videos on three realistic action datasets:



(a) (b) (c)

Figure 2: Mean Average Precision of our method with different numbers of centroids on (a) Hollywood2, (b) HMDB51 and (c) UCF101 using 50-NN as ground truth.

- **Hollywood2** [13]: A set of 2517 action samples spanning 12 action classes. All the videos are extracted from 69 different Hollywood movies. We use a clean subset of 1707 samples in our experiments.

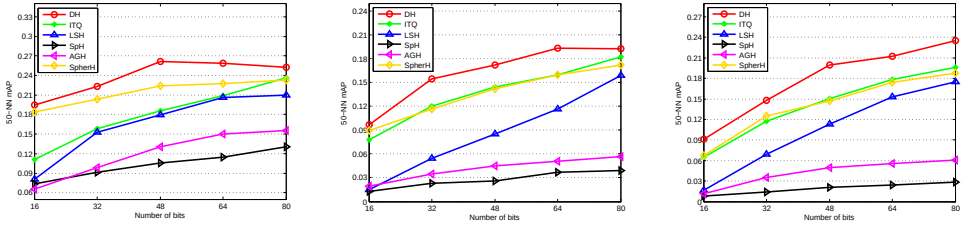
- **HMDB51** [6]: A large video database for human motion recognition, containing 6849 video clips of 51 categories collected from various sources such as movies and YouTube. We use the original videos rather than the stabilized ones in our experiments.

- **UCF101** [16]: A set of 13320 videos from 101 action categories. This gives the largest diversity in terms of actions and is one of the most challenging action datasets to date.

In terms of video representation, we follow [6, 13, 16] and employ Harris3D to detect Space-Time Interest Points (STIPs) [7] using the implementation by [13] and Histogram of Oriented Gradients (HOG)/Histogram of Optical Flow (HOF) as feature descriptors. For each video sequence, HOG and HOF are computed for 3D video patches in the neighborhood of STIPs. Then, HOG and HOF are concatenated to form the 162-element feature descriptor for each STIP. Finally, the Bag of Words (BoW) model is adopted to obtain the representation for each video. Specifically, we randomly choose 100k STIPs to build the visual vocabulary using k-means clustering and each STIP is assigned a label corresponding to its closest visual word of the vocabulary. The frequency of occurrences of each word belonging to the video clip is calculated and normalized into a histogram, which is the final BoW representation. We choose the size of the vocabulary as 5000, 5000 and 4000 for Hollywood2, HMDB51 and UCF101, respectively, which yield good results on these datasets.

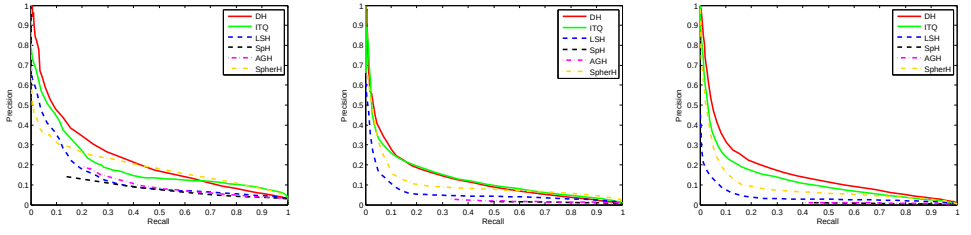
Our evaluation protocol follows those widely used in [2, 3, 10, 18, 19]. Specifically, 100, 200 and 500 samples are randomly chosen as queries for Hollywood2, HMDB51 and UCF101 respectively. The remaining samples form the training set against which queries are performed as well as the cross-validation set based on which parameters are learned. Queries are mutually exclusive with the training set. As in [3], k nearest neighbors (NN) of queries are used as ground truth based on the Euclidean distance. The mean Average Precision (mAP) is calculated based on the ground truth, and the precision-recall curve is drawn to measure the performance of our method. Due to the randomness of feature clustering, each experiment is performed 50 times and all the results reported in this paper are the averaged ones. After obtaining the video representations for three datasets, all the experiments for evaluating hashing algorithms are performed on a machine consisting of a 4-core i5 processor with 16GB main memory.

In addition, to obtain the similarity matrix $\mathbf{L}$, we need to cluster all the training samples to form K centroids. Figure 2 shows the mAP based on 10-fold cross-validation on the



(a) (b) (c)

Figure 3: Comparison with state-of-the-art methods in terms of mean Average Precision on (a) Hollywood2, (b) HMDB51 and (c) UCF101 using 50-NN as ground truth.



(a) (b) (c)

Figure 4: Comparison with state-of-the-art methods in terms of precision-recall curves@80 bits on (a) Hollywood2, (b) HMDB51 and (c) UCF101 using 50-NN as ground truth.

training sets of three datasets using different K. From Figure 2, we can see the variances of mAP with different K are very small on all the three datasets, which shows the performance of our method is not sensitive to the initial number of centroids. This further indicates the robustness of our hashing scheme. Specifically, K is set as 200 for all the three datasets.

3.2 Compared Methods

We compare our proposed DH method with five state-of-the-art hashing algorithms: Locality Sensitive Hashing (**LSH**) [4], Spectral Hashing (**SpH**) [20], Anchor Graphs Hashing with two-layer (**AGH**) [11], Iterative Quantization (**ITQ**) [2], and Spherical Hashing (**SpherH**) [3]. All of the above methods are evaluated on five different lengths of codes (*i.e.*, 16, 32, 48, 64 and 80). We use the codes provided by the authors and all the parameters used in the compared methods are strictly chosen according to their original papers.

3.3 Results on Hollywood2

Figure 3(a) shows the mean Average Precision of all the testing methods by using k Euclidean nearest neighbors as the ground truth when $k = 50$. The results are illustrated across different bit lengths from 16 to 80. It can be seen that our method, DH, performs better than the compared state-of-the-art methods in terms of all bit lengths. It is noteworthy that, when the bit length reaches 48, our method yields a 3.69% improvement over the second highest one. The results indicate that our method can approximate the similarities of original data points when small lengths of bits are assigned. This property is of vital importance for large-scale datasets, because a short binary code yielding reasonably high precision can always enhance the retrieval efficiency significantly. As shown in Figure 3(a), ITQ does not perform well

Methods	Precision(%)@top 5					Precision(%)@top 20				
	16bit	32bit	48bit	64bit	80bit	16bit	32bit	48bit	64bit	80bit
LSH	11.1	23.4	29.0	32.6	36.4	7.5	12.9	16.1	17.4	19.5
SpH	1.6	3.6	4.6	5.6	6.3	1.5	2.8	3.1	3.6	4.1
AGH	2.6	7.1	10.0	12.3	13.2	2.7	4.8	8.2	9.4	10.7
ITQ	10.9	18.9	23.5	26.0	30.4	8.4	12.7	15.1	16.6	18.0
SpherH	14.3	27.7	34.1	36.2	40.8	10.5	16.5	19.7	21.6	21.9
DH	**16.5**	**28.6**	**35.2**	**39.8**	**42.1**	**11.8**	**17.5**	**20.3**	**22.2**	**23.9**

Table 1: Results on UCF101 using semantic labels as ground truth.

when the number of bits is very small (*e.g.*, 16 and 32 bits), but when applied with more bits, its performance climbs up very quickly and is even better than SpherH. Additionally, we show precision-recall curves on Hollywood2 when allocating 80 bits in Figure 4(a), where similar performance gains of DH can be observed. From Figure 4(a), we can also make some interesting observations about the performance of SpH and AGH. The highest precisions of both methods are below 20%, which means a majority of returned points are non-relevant ones even when the recall rate is very low. In other words, close points in the Hamming space embedded by these two methods are not similar ones in the original Euclidean space.

3.4 Results on HMDB51

The mean Average Precision of different methods on HMDB51 are shown in Figure 3(b). The performance of our method climbs up quickly when the bit length increases from 16 to 64. However, the mAP decreases when the length continues to grow to 80, which indicates that 64 bits (*i.e.*, feature clusters) are more suitable for our method than 80 bits on HMDB51. ITQ and SpherH achieve similar performance on HMDB51, and the results of LSH turn better with the growth of bit length. Overall, the performance of DH is consistently better than all compared methods with all code lengths on HMDB51. Figure 4(b) shows the precision-recall curves by allocating 80 bits for all the testing methods, and similar results can be observed.

3.5 Results on UCF101

Figure 3(c) shows the mean Average Precision of all six methods on UCF101, which is the largest and most complex dataset in our experiments. As the number of bits increases, a majority of the methods increase rapidly in performance. However, as opposed to other methods, the performance of SpH and AGH increases very slowly, probably because these bit lengths are too short for them to achieve good performance on such a challenging dataset. Unlike on Hollywood2 and HMDB51, our DH improves notably in performance as the code length increases from 64 to 80. Due to the complexity of UCF101, a slightly longer code is needed to guarantee a good precision. Figure 4(c) shows the precision-recall curves when assigning 80 bits, which confirm the trends seen in Figure 3(c).

In addition, we evaluate the semantic consistency of codes generated by all the testing methods by using class labels as ground truth. In the semantic settings, the similarity matrix $\mathbf{L}$ is constructed based on the semantic labels of training data points and feature clustering is operated on the data centroids of the semantic classes. The average retrieval precisions of top 5 and 20 returned videos for each query are reported in Table 1 as in [2]. We can conclude

Bits	Runtime (ms) of Training (Coding)					
	LSH	SpH	AGH	ITQ	SpherH	**DH**
16	(87.0)	4451.8(119.4)	819.0(44.9)	5570.0(82.0)	4774.7(245.0)	1703.5(2.8)
32	(103.9)	4061.8(213.7)	843.9(43.1)	6093.3(114.6)	6756.2(244.3)	1538.7(2.7)
48	(130.6)	4513.7(420.2)	913.3(47.0)	6948.6(140.2)	7809.5(296.7)	1820.2(2.3)
64	(154.3)	4774.9(600.0)	857.7(45.3)	7936.2(159.6)	10139.0(316.9)	2125.3(1.8)
80	(167.9)	5202.6(875.4)	886.4(46.6)	7960.9(190.6)	13720.2(339.9)	2467.7(1.7)

Table 2: Comparison of the 50-run averaged runtime using different bits on UCF101.

that DH consistently outperforms other methods across all code lengths. This demonstrates that DH can also preserve the semantic consistency.

Finally, Table 2 illustrates the 50-run averaged training and coding time of different methods in terms of semantic hashing on UCF101. The numbers in brackets denote the coding time. Since LSH is based on random projection, its training time is not reported. ITQ and SpherH cost the most time for training since ITQ involves PCA as its first step, and both ITQ and SpherH are based on time-consuming iterative optimizations. Compared with most of other methods, our DH method needs much lower time complexity during training. Note that the total time cost of DH largely depends on the efficiency of feature clustering. Interestingly, we observe the cost of hash functions learning after feature clustering decreases as the code length increases. The reason is the dimension of each subspace becomes smaller when assigning more bits, and learning independent hash functions based on these subspaces will cost less time. As for coding, our DH method has the significant advantage over other methods, since most methods involve full-matrix projections.

4 Conclusion

In this paper, we have proposed a novel hashing scheme named Disaggregation Hashing for fast action retrieval from videos by incorporating feature clustering techniques. Disaggregation Hashing can address the hashing problem of high-dimensional (thousands or more) video data efficiently and effectively. The proposed method has been systematically evaluated on three realistic action datasets and the results demonstrate the superiority of our method over state-of-the-art hashing methods. In future work, it is worth considering employing more efficient feature clustering techniques and sophisticated hash functions to further enhance the performance.

Acknowledgements

This work was supported in part by the Hong Kong, Macao and Taiwan Science & Technology Cooperation Program of China (No. L2015TGA9004), the Fundamental Research Funds for the Central Universities, the China Scholarship Council, and Northumbria University.

References

[1] C. Ding and X. He. K-means clustering via principal component analysis. In *Proc. ICML*, 2004.

[2] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. CVPR*, 2011.

[3] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *Proc. CVPR*, 2012.

[4] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. STOC*, 1998.

[5] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010.

[6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proc. ICCV*, 2011.

[7] I. Laptev. On space-time interest points. *IJCV*, 64(2):107–123, September 2005.

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

[9] L. Liu and L. Wang. A scalable unsupervised feature merging approach to efficient dimensionality reduction of high-dimensional visual data. In *Proc. ICCV*, 2013.

[10] L. Liu, M. Yu, and L. Shao. Multiview alignment hashing for efficient image search. *IEEE TIP*, 24(3):956–966, March 2015.

[11] W. Liu, J. Wang, and S.-F. Chang. Hashing with graphs. In *Proc. ICML*, 2011.

[12] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. CVPR*, 2012.

[13] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *Proc. CVPR*, 2009.

[14] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007.

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

[16] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, November 2012.

[17] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *Proc. ICML*, 2010.

[18] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE TPAMI*, 34(12):2393–2406, December 2012.

[19] Q. Wang, G. Zhu, and Y. Yuan. Statistical quantization for similarity search. *CVIU*, 124(0):22–30, July 2014.

[20] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. NIPS*, 2009.

[21] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *Proc. NIPS*, 2001.