Binary Coding for Partial Action Analysis with Limited Observation Ratios

Jie Qin[1,2*], Li Liu[3,4*], Ling Shao[4], Bingbing Ni[5], Chen Chen[6], Fumin Shen[7] and Yunhong Wang[1,2†]
[1]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
[2]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
[3]Malong Technologies Co., Ltd., [4]University of East Anglia, [5]Shanghai Jiao Tong University
[6]Center for Research in Computer Vision, University of Central Florida
[7]University of Electronic Science and Technology of China
qinjiebuaa@gmail.com, li.liu@malongtech.cn, ling.shao@ieee.org, nibingbing@sjtu.edu.cn
chenchen870713@gmail.com, fumin.shen@gmail.com, yhwang@buaa.edu.cn

Abstract

Traditional action recognition methods aim to recognize actions with complete observations/executions. However, it is often difficult to capture fully executed actions due to occlusions, interruptions, etc. Meanwhile, action prediction/recognition in advance based on partial observations is essential for preventing the situation from deteriorating. Besides, fast spotting human activities using partially observed data is a critical ingredient for retrieval systems. Inspired by the recent success of data binarization in efficient retrieval/recognition, we propose a novel approach, named Partial Reconstructive Binary Coding (PRBC), for action analysis based on limited frame glimpses during any period of the complete execution. Specifically, we learn discriminative compact binary codes for partial actions via a joint learning framework, which collaboratively tackles feature reconstruction as well as binary coding. We obtain the solution to PRBC based on a discrete alternating iteration algorithm. Extensive experiments on four realistic action datasets in terms of three tasks (i.e., partial action retrieval, recognition and prediction) clearly show the superiority of PRBC over the state-of-the-art methods, along with significantly reduced memory load and computational costs during the online test.

1. Introduction

During the last decade, human action recognition [1, 43, 26, 28, 44, 5, 4, 40] has been extensively studied and most existing approaches aim to analyze after-the-fact actions, i.e., fully observed/executed actions (Fig. 1(a)). However, it is often too luxurious to capture complete actions, when devices or actions of interest are occluded, video transmissions are interrupted, etc. In these cases, we can only have partially observed actions and we refer to the problem as partial action recognition (*PAR*) (Fig. 1(c)). Particularly, if partial observations are only available from the beginning of full action executions, *PAR* becomes action prediction (*AP*) (Fig. 1(b)). Note that in this paper, we refer to a '**full/complete action**' as a video clip containing at least one complete action execution, and a '**partial/incomplete action**' means a shorter temporal segment of the video clip including incomplete observations of the complete action.

(a) (b) (c)

Figure 1. (a) Traditional action recognition/retrieval with full observations; (b) action prediction with partial observations at the beginning; (c) partial action recognition/retrieval with observations at **any** time (if partial observations are very **limited** and of **unknown** observation ratios, this turns into our problem).

Recognizing partial actions is essential for a wide range of applications. For instance, in surveillance systems, it is desirable to prevent potential harmful activities from happening by raising an alarm in advance based on partial observations. In smart homes, the robot needs to predict people's future activities and provide necessary services in time. In addition, content-based action retrieval systems can benefit a lot. With the help of *PAR*, there is no need to obtain complete actions before searching for their similar ones, which enhances efficiencies of retrieval systems a lot.

Several methods [41, 3, 18, 15, 11, 56, 61, 14, 10] have been proposed for *PAR* and *AP*, and most of them addressed

* indicates equal contributions.
† indicates corresponding author.

the latter one. For instance, Ryoo et al. [41] developed two variants of bag-of-words (BoW) representations, i.e., integral BoW and dynamic BoW, to identify unfinished activities from videos. Lan et al. [18] proposed a max-margin learning framework for action prediction based on a hierarchical representation. Kong et al. [15] formulated the task as a structured SVM learning problem and proposed the multiple temporal scale SVM (MTSSVM) for action prediction. A kernelized extension of MTSSVM was further introduced in [14].

Existing *PAR* methods usually require sufficient observations of complete actions to achieve acceptable performance. However, in most real-world cases, we should take actions immediately based on limited observations before the situation becomes worse, e.g., traffic accidents. And more observations will usually induce more memory usage and processing time. On the other hand, *AP* methods require partial observations available from the beginning of complete actions. However, video transmission interruptions or camera occlusions could happen at any time, which makes *AP* methods lack generality. Furthermore, most existing methods (e.g., [3, 15, 14]) hold an impractical assumption that observation ratios (ORs)[1] of partial actions are known during testing. Therefore, to overcome the above shortcomings, this paper aims at addressing a more general and practical case, where only a **short** temporal segment of **unknown** ORs, observed during **any** period of the complete execution is utilized for action analysis.

In addition, as we mentioned, *PAR* methods contribute a lot to video retrieval systems. However, existing methods are developed based on high-dimensional video data, whose *memory usage* and *computational costs* are unacceptable. For instance, if we represent 10 million videos using 4096-d features, the memory load is more than **300GB**, which is apparently infeasible for PCs or even workstations. Recently, learning-based binary coding/hashing methods [34, 8, 33, 12, 36, 30, 46, 31, 37, 27, 38, 35] have been exploited for large-scale image/video retrieval and classification. Based on these methods, high-dimensional video data can be embedded by short binary codes (e.g., we only need about **150MB** memory to load the data if embedded by 128-bit codes). Meanwhile, computational efficiency can be substantially improved since arithmetic operations are replaced by rapid **XOR** operations. Therefore, if we could develop the *PAR* method by incorporating the spirits of binary coding, our solution will be more scalable.

Among various hashing methods, there has been a series of deep learning based ones [23, 60, 59, 24]. Although they show promising results in retrieval tasks, they are not suitable for our problem due to the intrinsically inconsistency between queries (i.e., partial actions) and the database (i.e. full actions). Besides, cross-modality hashing (*CMH*) [2, 17, 58, 6, 22] (e.g., CVH [17] and SePH [22]) is mostly related to our problem. *CMH* performs binary coding for multimodal data (e.g., visual/text features). If we regard partial and full actions as data from two modalities, we may directly employ *CMH* to address *PAR*. Commonly, *CMH* learns either bidirectional projections or a joint projection for multi-modal data. In our problem, however, we are mostly interested in unidirectionally projecting partial data to the feature space of full data. Thus, *CMH* is not specially developed for our problem.

To address the aforementioned problems, we propose a novel approach to *learning discriminative **binary codes** for **partial** actions of **unknown** ORs during **any** period of the complete action execution*. We start with exploiting correlations between partial and full actions via an intuitive feature reconstruction method. After reconstruction, partial data can capture crucial discriminative information from full data. Subsequently, we incorporate the spirits of binary coding and discretely learn compact similarity-preserving binary codes. Various tasks can then be efficiently solved by matching the Hamming distances between binary codes of reconstructed partial test data and full training data.

The proposed method is motivated by the following two considerations. First, we would like to seek a 'reconstructed' binary representation for the partial action, therefore, we aim to minimize the reconstruction error between the reconstructed binary codes of partial actions and that of the corresponding full one. Second, we try to encourage 'discriminative capability' of the learned binary codes. As a common practice, we require that binary codes of the actions from the same category possess minimal distances while those from different classes are separated sufficiently. Moreover, to alleviate cumulative errors when addressing the above two considerations separately, we propose a novel joint learning framework, named Partial Reconstructive Binary Coding (PRBC), which is illustrated in Fig. 2. Our main **contributions** are three-fold:

1) We propose a novel binary coding approach for partial action analysis with limited observation ratios. Unlike *AP* methods, we can analyze partial actions observed during any period of complete executions. Moreover, our method significantly differs from existing *PAR* methods, since we can deal with partial actions with *unknown* and *very limited* ORs (typically *less than 30%*).

2) A joint learning framework, which collaboratively addresses feature reconstruction and binary coding, is proposed based on discrete alternating iteration. High-quality binary codes are learned without any relaxation. To our best knowledge, this is the first work proposing discrete binary coding techniques for partial action analysis in videos.

3) We present our approach in both *supervised* and *unsupervised* fashions and systematically evaluate it on four ac-

[1] Observation ratio: the ratio of the duration time t of the partial action to the duration time T of the corresponding full action, i.e., OR = $\frac{t}{T}$.

tion benchmarks in terms of three tasks, i.e., *partial action retrieval*, *recognition* and *prediction*. Compared to conventional methods, PRBC can achieve better accuracies with less *memory load* and fewer *testing computational costs*.

2. Partial Reconstructive Binary Coding

Given N full actions from some categories, the goal is to learn discriminative binary codes for any partial action and then perform partial action analysis (i.e., retrieval, recognition and prediction). The duration of a partial action is much shorter than that of a full action. Commonly, more observations contribute more to analyzing partial actions. Therefore, we first learn a feature reconstruction function that recovers partial actions to approximate the corresponding full ones. Secondly, we embed the reconstructed partial actions into compact binary codes. Finally, we couple the two problems and propose a joint optimization framework.

2.1. Feature Reconstruction for Partial Actions

We aim at learning a feature reconstruction function $\mathrm{g}(\cdot)$ that transfers crucial information from full data to partial data. Specifically, to learn this function, we choose M short temporal segments from each of N full actions to construct $M \times N$ corresponding partial actions.[2] After utilizing video representation techniques (e.g., local spatial temporal features [7, 19, 51] and global deep structural features [49, 13]), full and partial actions are represented by $\mathbf{y}_i \in \mathbb{R}^D$ and $\mathbf{x}_i^m \in \mathbb{R}^D$, respectively, where $i = 1, ..., N$ and $m = 1, ..., M$. $\mathbf{x}_i^m$ denotes the representation of the m-th partial action w.r.t. the i-th full action. The projection function $\mathrm{g}(\cdot)$ is learned as illustrated:

$$\mathbf{X} = [\mathbf{x}_1^1, \ldots, \mathbf{x}_1^M, \ldots, \mathbf{x}_i^1, \ldots, \mathbf{x}_i^m, \ldots, \mathbf{x}_i^M, \ldots, \mathbf{x}_N^1, \ldots, \mathbf{x}_N^M]$$
$$\Downarrow \mathrm{g}(\cdot)$$
$$\mathbf{Y} = [\underbrace{\mathbf{y}_1, \ldots, \mathbf{y}_1}_{M \text{ times}}, \ldots, \underbrace{\mathbf{y}_i, \quad \cdots \quad, \mathbf{y}_i}_{M \text{ times}}, \ldots, \underbrace{\mathbf{y}_N, \ldots, \mathbf{y}_N}_{M \text{ times}}]$$

Particularly, we define the feature reconstruction function as a linear projection function: $\mathrm{g}(\mathbf{x}_i^m) = \mathbf{W}^{\mathrm{T}}\mathbf{x}_i^m + \mathbf{c}$, where $\mathbf{W} \in \mathbb{R}^{D \times D}$ and $\mathbf{c} \in \mathbb{R}^D$ is the bias vector. If we denote $\widetilde{\mathbf{x}} = [\mathbf{x}^{\mathrm{T}}, 1]^{\mathrm{T}}$ and $\widetilde{\mathbf{W}} = [\mathbf{W}; \mathbf{c}^{\mathrm{T}}]$, it is equivalent to the projection without the bias. Therefore, we will omit $\mathbf{c}$ and the projection thus becomes $\mathrm{g}(\mathbf{x}_i^m) = \mathbf{W}^{\mathrm{T}}\mathbf{x}_i^m$.

After reconstruction, $\mathrm{g}(\mathbf{x}_i^m)$ should be as close as possible to the corresponding full data $\mathbf{y}_i$. To this end, we introduce a least-squares style objective function as follows:

$$\min_{\mathbf{W}} \sum_{i=1}^{N}\sum_{m=1}^{M} ||\mathbf{y}_i - \mathrm{g}(\mathbf{x}_i^m)||_2^2 = \sum_{i=1}^{N}\sum_{m=1}^{M} ||\mathbf{y}_i - \mathbf{W}^{\mathrm{T}}\mathbf{x}_i^m||_2^2. \quad (1)$$

Formula (1) may seem too strict since it enforces all reconstructed partial actions (including those with very limited motions) to approximate full ones. One may prefer more sophisticated schemes (e.g., multiple instance learning) for reconstruction. However, we find in our experiments that this simple formulation can fulfill the task well enough.

Without any additional terms, (1) can be explicitly solved. However, the solution is trivial since no supervision (i.e., semantic labels) is leveraged. We will consider this while introducing the following binary coding problem.

2.2. Discrete Binary Coding

We employ the widely-adopted sign function to obtain the L-bit binary codes $\mathbf{b}_i^m$ for the reconstructed partial data point $\mathbf{x}_i^m$, i.e., $\mathbf{b}_i^m = \mathrm{h}(\mathrm{g}(\mathbf{x}_i^m)) = \mathrm{sign}(\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x}_i^m))$, where the '$\mathrm{sign}(\cdot)$' function returns '1' if the argument is positive and '-1' otherwise, and $\mathbf{P} \in \mathbb{R}^{D \times L}$ is the coding matrix.

Similar to most learning-based hashing methods, we aim at preserving similarities between data points based on their semantic labels. In other words, a good binary code is expected to map points from the same class in the original space to similar binary codes in the Hamming space. Thus, we have the following objective function:

$$\min_{\mathbf{B},\mathbf{P}} \sum_{i,j=1}^{N}\sum_{m,n=1}^{M} s_{i,j}^{m,n}\, \mathrm{d}_h(\mathbf{b}_i^m,\, \mathbf{b}_j^n),\ \text{s.t. } \mathbf{b} = \mathrm{sign}(\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x})), \quad (2)$$

where $\mathbf{B} = \{\mathbf{b}_i^m\}$, $i = 1, ..., N$ and $m = 1, ..., M$, d_h is the Hamming distance and the semantic affinity

$$s_{i,j}^{m,n} = \begin{cases} 1.5, & \text{if } i = j \text{ and } Label_i^m = Label_j^n, \\ 1, & \text{if } i \neq j \text{ and } Label_i^m = Label_j^n, \\ -1, & \text{otherwise,} \end{cases} \quad (3)$$

where $Label_i^m$ and $Label_j^n$ denote action classes for $\mathbf{x}_i^m$ and $\mathbf{x}_j^n$ respectively. In other words, $s_{i,j}^{m,n}$ is positive if $\mathbf{x}_i^m$ and $\mathbf{x}_j^n$ come from the same semantic action class, and otherwise $s_{i,j}^{m,n}$ is negative. We further adopt a larger value (i.e., 1.5) if partial actions $\mathbf{x}_i^m$ and $\mathbf{x}_j^n$ are from the same full action, since they are more correlated with each other.

Since the Hamming distance between binary codes can be derived from the squared Euclidean distance [22], (i.e., $\mathrm{d}_h(\mathbf{b}_i^m,\, \mathbf{b}_j^n) = \frac{1}{4}||\mathbf{b}_i^m - \mathbf{b}_j^n||_2^2$), problem (2) can be rewritten as more tractable for optimization:

$$\min_{\mathbf{B},\mathbf{P}} \sum_{i,j=1}^{N}\sum_{m,n=1}^{M} s_{i,j}^{m,n}\, ||\mathbf{b}_i^m - \mathbf{b}_j^n||_2^2, \mathrm{s.t.} \mathbf{b} = \mathrm{sign}(\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x})). \quad (4)$$

Generally, this problem is NP-hard due to the discrete nature of the '$\mathrm{sign}(\cdot)$' function. Most existing hashing methods (e.g., [53, 52, 47, 39]) obtain approximate solutions by simply relaxing the discrete constraint. This usually yields the sub-optimal solution and will lead to less effective performance especially when learning long codes. Inspired by the recent success of discrete hashing [45, 46, 32], we keep

[2] If the duration time T of a full action is long enough, i.e., $T \geqslant M \times t$ (where t is the duration time of segments/partial actions), we randomly choose segments without overlaps. Otherwise, we choose with overlaps.

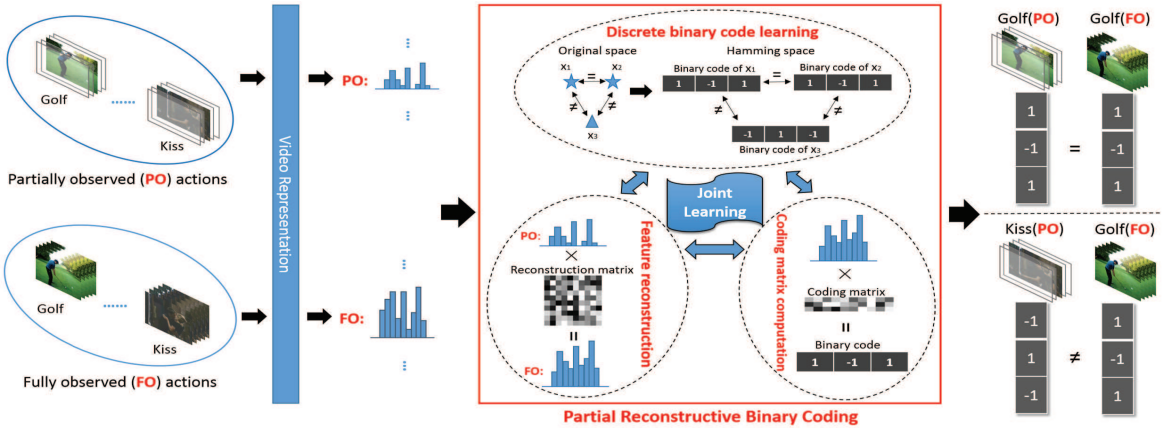

Figure 2. The overall framework of PRBC. We collaboratively learn a joint framework for feature reconstruction and binary coding. The learned binary codes are similarity-preserving and discriminative for action recognition/retrieval from partially observed actions.

the binary constraints in our problem and reformulate problem (4) as

$$\begin{aligned}\min_{\mathbf{B},\mathbf{P}} & \sum_{i,j=1}^{N}\sum_{m,n=1}^{M} s_{i,j}^{m,n}||\mathbf{b}_i^m-\mathbf{b}_j^n||_2^2 \\ & +\mu\sum_{i=1}^{N}\sum_{m=1}^{M}||\mathbf{b}_i^m-\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x}_i^m)||_2^2+\lambda||\mathbf{P}||_F^2 \\ \text{s.t. } & \mathbf{B}\in\{-1,1\}^{L\times MN},\end{aligned} \tag{5}$$

where $||\cdot||_F^2$ denotes the Frobenius norm, μ is the penalty parameter and λ is the regularization parameter. We pose an L2 regularizer on $\mathbf{P}$ to avoid overfitting and ensure numerical stability during coding. The penalty term models *the fitting error induced by the continuous function*. In practice, we can tolerate small differences between $\mathbf{b}$ and $\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x})$. Moreover, $\mathbf{P}$ can handle realistic *out-of-sample* problem.

2.3. Joint Optimization

Since $\mathbf{b}$ is a function of $\mathrm{g}(\mathbf{x})$, cumulative errors may occur if we consider feature construction and binary coding in isolation. In other words, errors induced by problem (1) will be further amplified after solving (5). Here, we propose Partial Reconstructive Binary Coding (PRBC) to collaboratively address problems (1) and (5). The joint objective function is defined as follows.

$$\begin{aligned}&\min_{\mathbf{B},\mathbf{W},\mathbf{P}}\sum_i\sum_m||\mathbf{y}_i-\mathrm{g}(\mathbf{x}_i^m)||_2^2+\sum_{i,j}\sum_{m,n}s_{i,j}^{m,n}||\mathbf{b}_i^m-\mathbf{b}_j^n||_2^2 \\ &\quad+\mu\sum_i\sum_m||\mathbf{b}_i^m-\mathbf{P}^{\mathrm{T}}\mathrm{g}(\mathbf{x}_i^m)||_2^2+\lambda||\mathbf{P}||_F^2 \\ &=\min_{\mathbf{B},\mathbf{W},\mathbf{P}}||\mathbf{Y}-\mathbf{W}^{\mathrm{T}}\mathbf{X}||_F^2+\sum_{i,j}\sum_{m,n}s_{i,j}^{m,n}||\mathbf{b}_i^m-\mathbf{b}_j^n||_2^2 \\ &\quad+\mu||\mathbf{B}-\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{X}||_F^2+\lambda||\mathbf{P}||_F^2 \\ &\text{s.t. } \mathbf{B}\in\{-1,1\}^{L\times MN},\end{aligned} \tag{6}$$

where $\mathbf{X},\ \mathbf{Y}\in\mathbb{R}^{D\times MN}$. Since the joint problem is non-convex, there is no global optimal solution. Here, we develop an alternating iteration algorithm to achieve the local optimum. Specifically, we iteratively optimize one variable while fixing the other two. In this way, we can achieve the local minimum of each variable. Since problem (6) is lower bounded, we can further guarantee the convergence of our method. Similar techniques are widely adopted in [46, 25, 45].

P-Step. By fixing $\mathbf{B}$ and $\mathbf{W}$, (6) is equivalent to the regularized least squares and $\mathbf{P}$ has a closed-form solution:

$$\mathbf{P}=(\mathrm{g}(\mathbf{X})\mathrm{g}(\mathbf{X})^{\mathrm{T}}+\frac{\lambda}{\mu}\mathbf{I}_D)^{-1}\mathrm{g}(\mathbf{X})\mathbf{B}^{\mathrm{T}}, \tag{7}$$

where $\mathrm{g}(\mathbf{X})=\mathbf{W}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{I}_D$ is a $D\times D$ identity matrix.

W-Step. With fixed $\mathbf{B}$ and $\mathbf{P}$, we obtain the solution to $\mathbf{W}$ by directly setting the derivative of (6) w.r.t. $\mathbf{W}$ to 0. Thus,

$$\mathbf{W}=(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}(\mathbf{X}\mathbf{Y}^{\mathrm{T}}+\mu\mathbf{X}\mathbf{B}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}})(\mathbf{I}_D+\mu\mathbf{P}\mathbf{P}^{\mathrm{T}})^{-1}. \tag{8}$$

B-Step. If all variables are fixed except $\mathbf{B}$, problem (6) turns into

$$\begin{aligned}&\min_{\mathbf{B}}\sum_{i,j}s_{i,j}||\mathbf{b}_i-\mathbf{b}_j||_2^2+\mu||\mathbf{B}-\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{X}||_F^2 \\ &\text{s.t. } \mathbf{B}\in\{-1,1\}^{L\times MN},\end{aligned} \tag{9}$$

where for simplicity, *we will omit m and n in problem (6) from now on, by setting $i,j=1,...,MN$*. To generate high-quality codes, we address the problem via a discrete coordinate descent algorithm. Specifically, we find a closed-form solution to one column $\mathbf{b}_i$ of $\mathbf{B}$ with all the other columns fixed. In other words, we iteratively learn the binary codes for each data point. Since $\{\mathbf{b}_j\}_{j\neq i}^{MN}$ are fixed, and $\mathbf{b}_i^{\mathrm{T}}\mathbf{b}_i=L\ (\forall i)$, the following equations hold:

$$\begin{aligned}&\min_{\mathbf{B}}\sum_{i,j}s_{i,j}||\mathbf{b}_i-\mathbf{b}_j||_2^2+\mu||\mathbf{B}-\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{X}||_F^2 \\ &=\min_{\mathbf{b}_i}\sum_{i,j}s_{i,j}||\mathbf{b}_i-\mathbf{b}_j||_2^2+\mu||\mathbf{b}_i-\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{x}_i||_2^2+const \\ &=\min_{\mathbf{b}_i}-2\mathbf{b}_i^{\mathrm{T}}(\sum_{j\neq i}s_{i,j}\mathbf{b}_j+\mu\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{x}_i)+const \\ &\text{s.t. } \mathbf{b}_i\in\{-1,1\}^L,\ i=1,...,MN.\end{aligned} \tag{10}$$

Algorithm 1: Anchor Approximation Strategy (AAS)

Input: $\widehat{\mathbf{B}} = \{\mathbf{b}_{a_k}\}_{k=1}^{K}, \{s_{i,j}\}_{i,j=1}^{MN}$.
Output: $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{MN}$.

for $i = 1 : MN$ **do**
  Find subset $\{\mathbf{x}_{a_j}\}_{j=1}^{\hat{K}}$ so that s_{i,a_j}=1;
  $j^* = \underset{j}{\operatorname{argmin}} ||\mathbf{x}_i - \mathbf{x}_{a_j}||_2^2, j = 1, ..., \hat{K}$;
  $\beta_i = a_{j^*}$;
end
return $\mathbf{B} = \{\mathbf{b}_{\beta_i}\}_{i=1}^{MN}$.

The above problem has the optimal discrete solution:

$$\begin{aligned}\mathbf{b}_i &= \operatorname{sign}(\sum_{j \neq i} s_{i,j}\mathbf{b}_j + \mu \mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{x}_i) \\ &= \operatorname{sign}(\mathbf{B}_{\neg i}\mathbf{s}_{i,\neg i} + \mu \mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{x}_i),\end{aligned} \tag{11}$$

where

$$\begin{cases}\mathbf{s}_{i,\neg i} = (s_{i,1}, ..., s_{i,i-1}, s_{i,i+1}, ..., s_{i,MN})^{\mathrm{T}} \\ \mathbf{B}_{\neg i} = [\mathbf{b}_1, ..., \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, ..., \mathbf{b}_{MN}],\end{cases} \tag{12}$$

i.e., $\mathbf{B}_{\neg i}$ indicates $\mathbf{B}$ excluding the i-th column $\mathbf{b}_i$. We can observe that computing the binary code for each data point relies on the rest pre-learned (MN-1) data points. Thus, we need to update $\mathbf{B}$ for MN times in the B-Step. Particularly, if we set the maximum iteration of our PRBC method as t, $\mathbf{B}$ should be updated for tMN times in total.

Anchor Approximation Strategy. Although optimal binary codes can be learned using all training points, we cannot guarantee the efficiency of the learning process when dealing with large-scale training set, i.e., large N. To this end, we propose the anchor approximation strategy (AAS). Specifically, some anchor points $\{\mathbf{x}_{a_k}\}_{k=1}^{K}$ ($K \ll N$) are randomly selected to learn the optimal codes $\widehat{\mathbf{B}} = \{\mathbf{b}_{a_k}\}_{k=1}^{K}$. We then approximate $\mathbf{B}$ with regard to $\widehat{\mathbf{B}}$, by reconstructing $\mathbf{B}$'s missing columns $\{\mathbf{b}_i\}_{i \neq a_k, \forall k}^{MN}$ based on the correlations between all points and selected anchor points. If $\mathbf{x}_i$ in all data points and $\mathbf{x}_{a_k}$ in anchor points are nearest neighbors with the same label, we approximate the binary code of $\mathbf{x}_i$ using that of $\mathbf{x}_{a_k}$. The AAS is illustrated in Algorithm 1, where β_i can be pre-computed. In each iteration, the approximation of $\mathbf{B}$ can operate in constant time. Moreover, to verify the effectiveness of AAS, we show some action retrieval results on UCF101 [48] in Fig. 3. Unsurprisingly, better performance is achieved using more anchor points. However, training time climbs up dramatically with increasing code lengths if we use all training data without AAS. Furthermore, based on 5,000∼8,000 anchor points, we can achieve competitive performance as compared with using all training data.

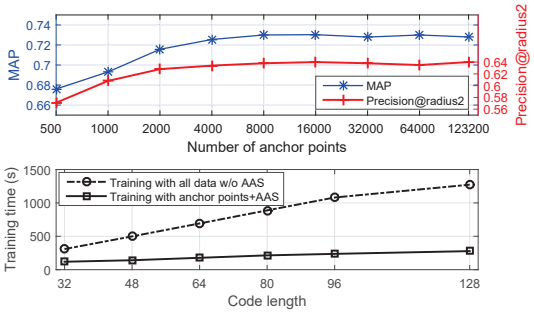

Figure 3. Evaluation of AAS on UCF101. The Mean Average Precision (MAP) and precision within Hamming radius 2 are reported using 48-bit codes. The total number of training points is 123,200.

Algorithm 2: Partial Reconstructive Binary Coding

Input: MN pairs of training data points $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{MN}$ w.r.t. partially and fully observed actions; semantic affinities $\{s_{i,j}\}_{i,j=1}^{MN}$; number of anchor points K; code length L; maximum iteration t; parameters μ and λ.
Output: Binary codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{MN} \in \{-1, 1\}^{L \times MN}$; feature reconstruction function $\mathrm{g}(\mathbf{x}) = \mathbf{W}^{\mathrm{T}}\mathbf{x}$; binary coding function $\mathrm{h}(\mathbf{x}) = \operatorname{sign}(\mathbf{P}^{\mathrm{T}}\mathbf{x})$.

Randomly select K pairs of data points $\{\mathbf{x}_{a_k}, \mathbf{y}_{a_k}\}_{k=1}^{K}$; randomly initialize $\widehat{\mathbf{B}} = \{\mathbf{b}_{a_k}\} \in \{-1, 1\}^{L \times K}$; initialize $\mathbf{W} = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}(\mathbf{X}\mathbf{Y}^{\mathrm{T}})$ and $\mathbf{P}$ using Eq. (7);
Loop until convergence or reach t iterations:
  - B-Step: Update $\widehat{\mathbf{B}}$ using Eq. (11); Approximate $\mathbf{B}$ using Algorithm 1;
  - W-Step: Update $\mathbf{W}$ using Eq. (8);
  - P-Step: Update $\mathbf{P}$ using Eq. (7).

The overall PRBC method is summarized in Algorithm 2. Through our experiments, we find that PRBC can successfully converge within $t = 3 \sim 5$ iterations.

Unsupervised Learning. Although we cannot obtain the semantic affinities $s_{i,j}$ explicitly in the unsupervised setting, we can employ pseudo affinities based on the posteriori smoothness assumption [29, 38]. Specifically, we first employ k-means clustering to obtain data clusters based on the full training set and $s_{i,j} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster, and $s_{i,j} = -1$ otherwise. In this way, our method is developed in both supervised and unsupervised fashions, leading to a more general solution.

Online Test. Once the joint learning is finished, we can obtain the optimal $\mathbf{W}$ and $\mathbf{P}$. In the test phase, when a novel partial action $\mathbf{x}_{part}$ comes, we first recover its representation by applying the feature reconstruction function, i.e., $\mathrm{g}(\mathbf{x}_{part}) = \mathbf{W}^{\mathrm{T}}\mathbf{x}_{part}$. Then we utilize our binary coding function to obtain the binary representation, i.e., $\mathbf{b}_{part} = \mathrm{h}(\mathrm{g}(\mathbf{x}_{part})) = \operatorname{sign}(\mathbf{P}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{x}_{part})$. As for any full action $\mathbf{y}_{full}$, we directly apply the sign function to obtain its binary code, i.e., $\mathbf{b}_{full} = \operatorname{sign}(\mathbf{P}^{\mathrm{T}}\mathbf{y}_{full})$. Various action analysis tasks can then be efficiently solved by matching the Hamming distances between $\mathbf{b}_{part}$ and $\mathbf{b}_{full}$.

Table 1. Comparison of action retrieval performance on HMDB51 w.r.t. 16-frame partial actions using 128-bit binary codes.

Method			MAP (%)	Precision @radius2 (%)	Precision @rank50 (%)	Training time (s)	Test coding time (s)
Single-Modality Binary Coding Methods	Supervised	SDH [46]	29.80	0.11	37.92	477.17	3.8×10^{-6}
		FastHash [21]	47.46	0.001	54.05	1.56×10^{3}	9.3×10^{-4}
		KSH [34]	5.10	0.095	8.77	1.17×10^{3}	3.4×10^{-6}
		CCA-ITQ [8]	34.71	2.58	42.61	8.90	2.9×10^{-6}
	Unsupervised	AGH [33]	3.08	1.27	2.10	35.58	3.8×10^{-6}
		PCA-ITQ [8]	2.94	<0.001	2.44	7.62	4.4×10^{-6}
Cross-Modality Binary Coding Methods	Supervised	SePH [22]	50.20	23.42	54.18	2.14×10^{3}	3.7×10^{-6}
		SCM [58]	37.14	3.11	43.62	204.52	8.5×10^{-6}
		CVH [17]	14.41	1.54	24.98	25.21	2.2×10^{-6}
		CMSSH [2]	35.87	1.85	37.85	895.75	6.4×10^{-6}
	Unsupervised	CMFH [6]	5.02	2.42	3.93	411.37	7.3×10^{-6}
Proposed	Supervised	**PRBC-Sup**	**59.71**±0.754	**32.31**±0.521	**63.24**±0.630	129.01	3.4×10^{-6}
	Unsupervised	**PRBC-Unsup**	32.27±0.717	16.94±0.448	39.80±0.692	144.34	3.2×10^{-6}
4096-d C3D Feature (CF)			2.91	-	2.01	-	-
4096-d C3D Feature+Reconstruction (CF+R)			12.4	-	10.76	129.01	-

'**C3D Feature (CF)**' means we conduct experiments using original C3D features of full training data ($\mathbf{y}$) and partial test data ($\mathbf{x}$) as the database and queries, respectively. '**C3D Feature+Reconstruction (CF+R)**' means we employ the reconstructed features of test data $\mathbf{W}^{\mathrm{T}}\mathbf{x}$ as queries. The standard deviations of our methods for 10 runs are also reported. The **memory load** for 4096-d C3D features and 128-bit binary codes are around **180MB** and **90KB**, respectively. The **computational costs** for calculating the Euclidean/Hamming distances between two features/codes are approximately $\mathbf{1\times10^{-3}}$ and $\mathbf{5\times10^{-7}}$ seconds.

3. Experiments and Results

We conduct extensive experiments on realistic action datasets in terms of three tasks, i.e., partial action retrieval, recognition and prediction. As *partial action retrieval* and *recognition* share similar experimental setup, we will introduce it in the following. As for *action prediction*, we will elaborate its setting in Section 3.3.

Datasets. Our approach is evaluated on two large-scale realistic action datasets, i.e., HMDB51 [16] and UCF101 [48]. HMDB51 is known as a large video database for human action recognition, containing 6,766 actions from 51 categories collected from various sources. UCF101 contains a set of 13,320 videos from 101 categories. This gives the largest diversity in terms of actions and is one of the most challenging action datasets to date.

Action Representation. We adopt the deep neural networks namely C3D [49] for spatial-temporal feature extraction. We strictly follow [49] and split actions into 16-frame segments with an overlap of 8 frames. The fc6-layer activations are extracted as 4096-d features for each segment. A full/partial action is then represented by a 4096-d descriptor, which is the averaged and L2-normalized feature of all the segments in the full/partial action.

Construction of Partial Actions. Since we employ C3D to extract features per 16-frame segment, we simply simulate partial actions using 16/32-frame segments in our experiments. During training, we randomly select M 16-frame or 32-frame segments from each training full action. Therefore, there are a total of $M\times N$ pairs of partially and fully observed actions regarding the 16/32-frame setting, where N is the size of the training set. These segments are also utilized for learning the feature reconstruction function.

Remark. **1)** As we randomly choose the segments, partial actions are guaranteed to be observed during **any** time of full actions. **2)** Although we choose fixed 16/32-frame segments as partial actions, they are still of **unknown** ORs since full actions have various numbers of frames. This makes our method unique from other counterparts [3, 15, 14] which require ORs to be known during testing. **3)** Since full actions on the two datasets have around 140 frames on average, it is reasonable to utilize 16/32-frame segments as partial actions which meet the requirement of **limited** ORs (i.e., less than 30%).

Protocol. **1)** In terms of *partial action retrieval*, we randomly choose 1,000 actions from each dataset for testing. We also randomly choose M 16/32-frame segments from each test full action as partial actions. The $1000M$ partial actions are used as queries regarding the 16/32-frame setting. The rest N full actions form the training set as well as the retrieval database. Queries are performed against all full data from the database. Based on Fig. 3, we randomly choose 5,000 pairs from the training set, to achieve good performance and ensure high computational efficiency. We learn the optimized reconstruction and binary coding functions based on these anchor pairs and the corresponding semantic affinities. Parameters are tuned via a cross-validation on the training set. Specifically, we set $M = 10$, $\mu = 10$, $\lambda = 0.01$ and $t = 5$. Semantic action labels are utilized as ground truth. Similar to other hashing methods, experimental results in terms of both hash lookup (precision) and Hamming ranking (MAP) are reported to evaluate performance. Note that we conduct the experiments in both supervised and unsupervised settings, which correspond to PRBC-Sup and PRBC-Unsup, respectively. For the unsupervised setting, we obtain the affinities $s_{i,j}$ by k-means clustering on the training set with empirical $k = \sqrt{N}$ [50, 38]. Due to the randomness of choosing test actions, we report the average performance via 10 runs. **2)** As for *partial action recognition*, most protocols are equivalent to those of partial action retrieval, except that we need to predict labels for 16/32-frame test partial actions rather than find their similar actions. Therefore, a linear SVM classifier is learned based on binary codes of full training data

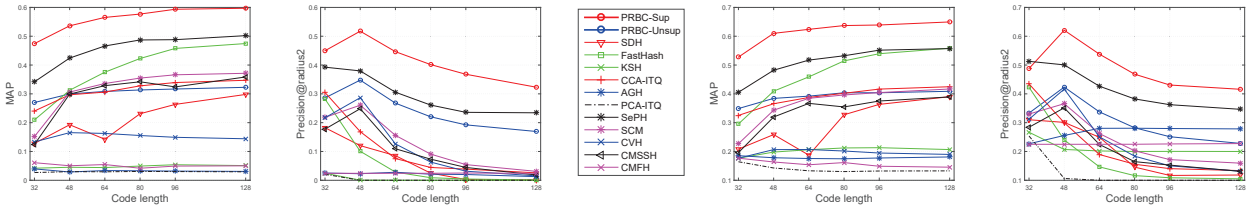

Figure 4. Retrieval results on HMDB51 using 16-frame (left two figures) and 32-frame (right two figures) partial actions as queries.

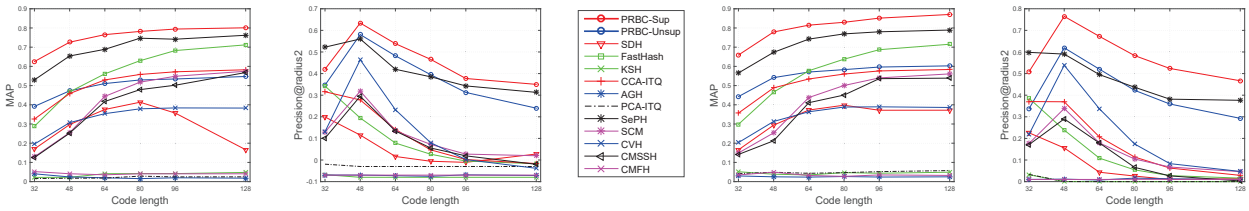

Figure 5. Retrieval results on UCF101 using 16-frame (left two figures) and 32-frame (right two figures) partial actions as queries.

and then applied to binary codes of partial test data for final recognition. We conduct the experiments on a PC with an Intel quad-core 3.4GHz CPU and 32GB memory.

Compared Methods. Since PRBC is a data binarization method, we compare it with numerous state-of-the-art binary coding methods. Single-modality hashing methods include supervised ones (i.e., SDH [46], FastHash [21], KSH [34] and CCA-ITQ [8]), and unsupervised ones (i.e., AGH [33] and PCA-ITQ [8]). We also employ several cross-modality hashing to respectively learn projections for partial actions and full ones, since cross-modality hashing is more intrinsically similar to our solution. Cross-modality supervised hashing methods include SePH [22], SCM with the sequential learning [58], CVH [17] and CMSSH [2]. CMFH [6] is an unsupervised cross-modality method. For fair comparisons, we stack both full and partial data with the corresponding labels into the whole training data, which is then utilized in the training phase of single-modality methods. We use public codes of all methods except CVH, which is implemented by following [17]. Parameters of compared methods are optimized for best performance, which facilitates fair comparisons with ours.

3.1. Partial Action Retrieval

We first test our method on HMDB51 in terms of effectiveness and efficiency. Table 1 shows the results using 16-frame partial actions as queries with 128-bit codes. Generally, cross-modality methods outperform single-modality ones and supervised methods perform better than unsupervised ones. PRBC-Sup clearly outperforms all other methods. PRBC-Unsup also has significant advantages over other unsupervised methods and even outperforms several supervised ones. We also show the results using 'C3D Feature (CF)' and 'C3D Feature+Reconstruction (CF+R)'. The performance of 'CF' is very poor due to significant inconsistency between full and partial actions. By feature reconstruction using $\mathbf{W}$, the performance can be enhanced a lot, which proves the effectiveness of reconstruction. However, 'CF+R' still cannot perform as well as PRBC because of lacking supervised information. Regarding efficiency, our method requires less training time in most cases, owing to our AAS. As for the online coding time, our methods are comparable to others but much faster than FastHash.

We show the results on HMDB51 across six code lengths in Fig. 4. Most methods can achieve better performance with more observations from 16 to 32 frames. Notably, PRBC-Sup consistently outperforms all compared methods regardless of code lengths. PRBC-Unsup also shows the superiority over other unsupervised methods and even works better than several supervised ones (e.g., CVH and SDH).

Fig. 5 shows results of different methods on UCF101. Enhanced results can be obtained with increasing bits, and the improvement is obvious from 32 to 48. This indicates extreme short codes may lack discrimination and are unsuitable for large-scale complex datasets. Generally, PRBC-Sup achieves the best performance among all other methods and PRBC-Unsup also shows its advantage over other unsupervised methods and several supervised ones.

3.2. Partial Action Recognition

Since our learned binary codes can be regarded as compact features, we evaluate the effectiveness of PRBC regarding partial action recognition. For both datasets, we follow the standard 3 splits setting [16, 48] and report the average recognition accuracy. The comparison results across three code lengths are shown in Table 2. With increasing bits, most methods can obtain more accurate results. Our PRBC-Sup performs the best across all code lengths. PRBC-Unsup also achieves promising results. With only 32 bits, our method can already obtain satisfactory results, i.e., more than 40% and 70% accuracies on HMDB51 and UCF101 respectively. We also compare PRBC with several

Table 2. Partial action recognition accuracies (%) of different methods on HMDB51 and UCF101.

Method		16-frame partial actions for testing						32-frame partial actions for testing					
		HMDB51			UCF101			HMDB51			UCF101		
		32 bits	64 bits	128 bits	32 bits	64 bits	128 bits	32 bits	64 bits	128 bits	32 bits	64 bits	128 bits
Single-Modality Binary Coding Methods	SDH [46]	13.91	16.47	19.36	27.63	38.05	44.78	12.31	15.15	19.35	33.06	42.78	50.33
	FastHash [21]	16.70	21.08	23.09	37.17	48.57	55.98	15.15	18.14	20.11	39.51	49.31	56.29
	CCA-ITQ [8]	17.45	19.03	21.23	49.23	52.59	54.90	19.42	20.80	22.63	52.53	56.77	59.57
	KSH [34]	2.23	2.85	2.62	2.87	2.28	2.47	2.58	2.73	2.31	7.98	8.02	8.85
	AGH [33]	5.36	6.20	5.5	1.97	1.94	1.47	3.33	3.33	4.62	3.74	4.40	3.82
	PCA-ITQ [8]	2.30	3.08	3.22	4.74	4.94	4.74	4.02	3.63	3.87	6.32	7.39	7.11
Cross-Modality Binary Coding Methods	SePH [22]	32.89	33.97	37.15	57.61	63.61	67.84	37.07	39.58	41.24	59.21	65.06	69.11
	SCM [58]	31.78	36.48	38.67	40.94	62.06	68.57	31.57	35.75	37.95	41.03	62.29	68.97
	CVH [17]	25.52	31.13	34.93	45.07	56.78	64.70	26.32	31.55	36.04	45.90	57.92	66.17
	CMFH [6]	2.65	2.60	3.15	7.04	7.32	7.95	3.94	3.85	4.91	8.84	8.42	9.53
Proposed	**PRBC-Sup**	**42.78**	**45.80**	**48.60**	**70.27**	**75.11**	**78.46**	**46.52**	**49.32**	**50.79**	**71.79**	**77.47**	**80.80**
	PRBC-Unsup	29.64	32.76	34.25	58.06	62.94	67.15	31.64	33.90	34.84	56.15	60.49	64.19
Cross-View Feature Learning Methods	CCA* [9]	39.51 (2048-d)			70.61 (4096-d)			41.87 (2048-d)			72.26 (4096-d)		
	PLSR* [54]	37.71 (4096-d)			66.83 (4096-d)			40.02 (4096-d)			68.05 (4096-d)		
	XQDA* [20]	11.53 (512-d)			40.11 (512-d)			14.32 (512-d)			44.14 (512-d)		
	CVFL [55]	40.32 (4096-d)			70.97 (4096-d)			44.12 (4096-d)			73.13 (4096-d)		
C3D Feature (CF)		3.42 (4096-d)			3.92 (4096-d)			4.70 (4096-d)			4.93 (4096-d)		
C3D Feature + Reconstruction (CF+R)		24.39 (4096-d)			53.50 (4096-d)			30.09 (4096-d)			55.12 (4096-d)		

'*' indicates that we test these methods using feature subspaces of different dimensions, ranging from 512-d to 4096-d with step-size 512, and show the best results here. Number in the bracket denotes the feature dimension w.r.t. the result. Similar to '**CF+R**', **CVFL** reconstructs partial data to approximate full data, so its dimension is fixed to 4096.

cross-view feature learning methods (i.e., CCA[3] [9], PLSR [54], XQDA [20] and CVFL [55]) by treating full/partial actions as different views. PRBC can even outperform state-of-the-art cross-view learning methods. Although PRBC-Sup with 32 bits performs slightly worse than CVFL/CCA with 4096-d ($\approx 2.6\times10^5$-bit) features, it reduces memory load and computational costs significantly.

3.3. Action Prediction

Since *AP* is a special case of *PAR*, we also evaluate our method in the context of *AP*. We employ the widely used dataset: UT-Interaction [42], which contains two subsets and each one has 6 classes of high-level interactions. As there are only 60 action videos in each subset, we learn PRBC from all training data instead of using AAS. We follow the standard experimental setting [41]. Specifically, we adopt the Cuboids descriptors [7] and employ BoW with 800 codewords. Following [41], we adopt the leave-one-sequence-out scheme, i.e., 10-fold cross validation for each subset. The average prediction accuracy is reported regarding different ORs. In particular, since we are more interested in predicting actions with limited observations, we evaluate all methods with three small ORs.

Table 3 shows the prediction accuracies. All the compared results are the best ones reported in the original papers. With increasing ORs, all methods achieve better performance, which proves our assumption that more observations contribute more to action analysis. In almost all cases, our method achieves the best accuracies. Performance gains are especially obvious with small ORs, showing the effectiveness of our method in handling partial actions with very limited observations. It is worth noting that all compared methods employ the original 800-d ($\approx 5\times10^4$-bit) features, while we reduce them into 32/64-bit binary codes. This further demonstrates the superiority of the proposed PRBC.

[3]CCA regards partial/full actions as two views to learn the common real-valued subspace, while CCA-ITQ utilizes actions and their labels to learn binary codes.

Table 3. Action prediction accuracies (%) on UT-Interaction dataset #1 and #2 w.r.t. different observation ratios (ORs).

Method	UT-Interaction dataset #1			UT-Interaction dataset #2		
	OR=0.1	OR=0.2	OR=0.3	OR=0.1	OR=0.2	OR=0.3
Bayesian [41]	16.7	16.7	16.7	16.7	16.7	17.1
BP-SVM [41]	16.8	21.7	27.8	16.7	24.0	35.5
IBoW [41]	14.5	17.9	30.8	16.8	29.9	34.9
DBoW [41]	15.2	20.2	30.7	16.7	28.9	43.3
SC [3]	18.3	33.3	56.7	21.7	43.3	50.0
MSSC [3]	18.3	40.0	60.0	21.7	40.0	48.3
MTSSVM [15]	36.7	46.7	66.7	33.3	50.0	60.0
RPT [57]	13.3	26.7	56.7	15.0	33.3	63.3
AAC [56]	45.0	46.7	60.0	51.3	53.3	60.0
MOVEMES [18]	38.3	54.5	68.3	31.3	41.3	56.7
MMAPM [14]	46.7	51.7	**70.0**	36.7	55.0	63.3
PRBC-Sup@64bits	55.0	**58.3**	63.3	**60.0**	**65.0**	**75.0**
PRBC-Sup@128bits	**56.7**	**58.3**	65.0	**60.0**	63.3	71.7

4. Conclusion

In this paper, we proposed a novel approach, named Partial Reconstructive Binary Coding (PRBC), for tackling efficient partial action analysis with *limited ratios of observations*. A joint learning framework was developed for feature reconstruction and discrete optimization of discriminative binary codes for partial actions. We systematically evaluated the proposed method regarding three tasks, i.e., partial action retrieval, recognition and prediction on four public datasets. The results consistently demonstrated the superiority of our method over the state-of-the-arts in terms of both accuracy and storage/computation efficiency.

Acknowledgement

This work was partly supported by the National Natural Science Foundation of China (No. 61573045), the Foundation for Innovative Research Groups through the National Natural Science Foundation of China (No. 61421003), the National Natural Science Foundation of China (No. 61502301), and China's Thousand Youth Talents Plan.

References

[1] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *CSUR*, 43(3):16:1–16:43, 2011.

[2] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, 2010.

[3] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. M. Siskind, and S. Wang. Recognize human activities from partially observed videos. In *CVPR*, 2013.

[4] C. Chen, R. Jafari, and N. Kehtarnavaz. Improving human action recognition using fusion of depth camera and inertial sensors. *IEEE Transactions on Human-Machine Systems*, 45(1):51–61, 2015.

[5] C. Chen, M. Liu, B. Zhang, J. Han, J. Jiang, and H. Liu. 3d action recognition using multi-temporal depth motion maps and fisher vector. In *IJCAI*, 2016.

[6] G. Ding, Y. Guo, and J. Zhou. Collective matrix factorization hashing for multimodal data. In *CVPR*, 2014.

[7] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

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

[9] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[10] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, and J. Lai. Real-time rgb-d activity prediction by soft regression. In *ECCV*, 2016.

[11] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014.

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

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[14] Y. Kong and Y. Fu. Max-margin action prediction machine. *IEEE TPAMI*, 2015.

[15] Y. Kong, D. Kit, and Y. Fu. A discriminative model with multiple temporal scales for action prediction. In *ECCV*, 2014.

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

[17] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, 2011.

[18] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014.

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

[20] S. Liao, Y. Hu, X. Zhu, and S. Z. Li. Person re-identification by local maximal occurrence representation and metric learning. In *CVPR*, 2015.

[21] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[22] Z. Lin, G. Ding, M. Hu, and J. Wang. Semantics-preserving hashing for cross-view retrieval. In *CVPR*, 2015.

[23] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2016.

[24] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, 2016.

[25] L. Liu, Z. Lin, L. Shao, F. Shen, G. Ding, and J. Han. Sequential discrete hashing for scalable cross-modality similarity retrieval. *IEEE TIP*, 2016.

[26] L. Liu and L. Shao. Learning discriminative representations from rgb-d video data. In *IJCAI*, 2013.

[27] L. Liu and L. Shao. Sequential compact code learning for unsupervised image hashing. *IEEE TNNLS*, 27(12):2526–2536, Dec. 2016.

[28] L. Liu, L. Shao, X. Zhen, and X. Li. Learning discriminative key poses for action recognition. *IEEE TCYB*, 43(6):1860–1870, Dec. 2013.

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

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

[31] L. Liu, M. Yu, and L. Shao. Projection bank: From high-dimensional data to medium-length binary codes. In *ICCV*, 2015.

[32] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, 2014.

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

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

[35] X. Liu, X. Fan, C. Deng, Z. Li, H. Su, and D. Tao. Multilinear hyperplane hashing. In *CVPR*, 2016.

[36] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashing. In *CVPR*, 2014.

[37] X. Liu, L. Huang, C. Deng, J. Lu, and B. Lang. Multi-view complementary hash tables for nearest neighbor search. In *ICCV*, 2015.

[38] J. Qin, L. Liu, M. Yu, Y. Wang, and L. Shao. Fast action retrieval from videos via feature disaggregation. In *BMVC*, 2015.

[39] J. Qin, L. Liu, M. Yu, Y. Wang, and L. Shao. Fast action retrieval from videos via feature disaggregation. *CVIU*, 156:104–116, 2017.

[40] J. Qin, L. Liu, Z. Zhang, Y. Wang, and L. Shao. Compressive sequential learning for action similarity labeling. *IEEE TIP*, 25(2):756–769, 2016.

[41] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011.

[42] M. S. Ryoo and J. K. Aggarwal. UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA), 2010.

[43] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[44] L. Shao, L. Liu, and M. Yu. Kernelized multiview projection for robust action recognition. *IJCV*, 118(2):115–129, June 2016.

[45] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. Tao Shen. Learning binary codes for maximum inner product search. In *ICCV*, 2015.

[46] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, 2015.

[47] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *CVPR*, 2013.

[48] 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*, 2012.

[49] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[50] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011.

[51] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

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

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

[54] H. Wold. Partial least squares. *Encyclopedia of statistical sciences*, 6:581–591, 1985.

[55] W. Xie, Y. Peng, and J. Xiao. Cross-view feature learning for scalable social image analysis. In *AAAI*, 2014.

[56] Z. Xu, L. Qing, and J. Miao. Activity auto-completion: Predicting human activities from partial videos. In *ICCV*, 2015.

[57] G. Yu, J. Yuan, and Z. Liu. Propagative hough voting for human activity detection and recognition. *IEEE TCSVT*, 25(1):87–98, Jan 2015.

[58] D. Zhang and W.-J. Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI*, 2014.

[59] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *CVPR*, 2016.

[60] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[61] Y. Zhou and T. L. Berg. Temporal perception and prediction in ego-centric video. In *ICCV*, December 2015.