PAPER

Efficient Local Feature Encoding for Human Action Recognition with Approximate Sparse Coding

Yu WANG[†], *Nonmember* and Jien KATO^{†a)}, *Member*

SUMMARY Local spatio-temporal features are popular in the human action recognition task. In practice, they are usually coupled with a feature encoding approach, which helps to obtain the video-level vector representations that can be used in learning and recognition. In this paper, we present an efficient local feature encoding approach, which is called Approximate Sparse Coding (ASC). ASC computes the sparse codes for a large collection of prototype local feature descriptors in the off-line learning phase using Sparse Coding (SC) and look up the nearest prototype's precomputed sparse code for each to-be-encoded local feature in the encoding phase using Approximate Nearest Neighbour (ANN) search. It shares the low dimensionality of SC and the high speed of ANN, which are both desired properties for a local feature encoding approach. ASC has been extensively evaluated on the KTH dataset and the HMDB51 dataset. We confirmed that it is able to encode large quantity of local video features into discriminative low dimensional representations efficiently.

key words: approximate sparse coding, sparse coding, approximate nearest neighbour, local feature encoding, action recognition

1. Introduction

Visual recognition of human action is an important capability of computational intelligence. It serves as a key component in many applications, such as intelligent video surveillance, automatic content annotation, human computer interaction and so on. Typical methods for this task follow the conventional pattern recognition paradigm, which consists of two steps including feature encoding and model training. Over the last decade, many features and models have been proposed in the literature.

In particular, the use of local spatio-temporal features has drawn a great deal of attentions and shown promising performance. Local spatio-temporal features, such like Space-time Interest Point (STIP) features [1] and Improved Dense Trajectory (IDT) features [2], [3], are able to capture the local characteristics of videos while preserving strong robustness against viewpoint changes and background clutters. They are usually utilized in the bag-of-words (BOW) [4] framework, and have proved to be very powerful in the human action recognition task [5]. In practice, when using such local features, it is critical to cope them with an appropriate encoding approach, which helps to obtain a discriminative video-level vector representation.

In the literature, there are many successful encod-

ing approaches, including Vector Quantization (VQ) [4], Sparse Coding (SC) [6], Locality-constrained Linear Coding (LLC) [7], Vector of Locally Aggregated Descriptors (VLAD) [8], Fisher Vector (FV) [9] and so on. These approaches can be coarsely divided into two categories according to whether the codes are sparse vectors [4], [6], [7] or super vectors [8]–[10]. Assume a codebook of size k is used to encode a d dimensional local feature, the codes of approaches in the first category are k dimensional sparse vectors. On the other hand, the codes of approaches in the second category are odk dimensional super vectors, where $o = 1$ if only the first order statistics are embedded [8] and $o = 2$ if both the first and the second order statistics are embedded [9]. Video-level vector representations are then obtained by pooling the codes of features within the whole space-time volume. The dimensionality is determined by the encoding approach.

In recent works, super vector based encoding approaches have shown good performance and outperform sparse vector based approaches a significant margin in many benchmark datasets [11]. Though they perform well, an obvious issue of these approaches is that the dimensionality of the final video-level vector representation is usually very high. For example, in [3] the FVs are 109,056 dimensional and in [12] the combined FVs are 204,800 dimensional. Such high dimensional vectors not only put the pyramid based video representations [13], [14] out of the table, but also make even linear Support Vector Machines (SVMs) become difficult to train. Furthermore, conventional dimension reduction approaches, such like Principal Component Analysis (PCA), are not possible to be utilized because the projection matrices are too large to be manipulated in the main memory of current generation personal computers.

In contrast, sparse vector based encoding approaches enjoy a very good property of low dimensionality, but suffers from a high computational cost, especially for encoding of video features. The reasons are: (1) to obtain informative representations, these approaches are usually implemented with a large codebook (k is usually set to 4,000 in many works [5]); (2) the number of local features of videos is usually very large (a 10 seconds video in the HMDB51 dataset [15] may output more than 100,000 IDT features). Since most sparse vector based encoding approaches require the pair-wise distance between the to-be-encoded local feature and all words in the codebook to be computed, coding a video is very time consuming even for the most traditional VQ approach. This issue is rather obvious for SC, which

Manuscript received August 21, 2015.

Manuscript revised November 30, 2015.

Manuscript publicized January 6, 2016.

[†]The authors are with Nagoya University, Nagoya-shi, 464-8603 Japan.

a) E-mail: jien@is.nagoya-u.ac.jp

DOI: 10.1587/transinf.2015EDP7333

has to solve a l_1 -norm regularized least squares problem for each descriptor. As discussed in [16], a full evaluation of SC on a dataset such like HMDB51 [15] could take months, which makes SC impracticable for encoding video features.

In this paper, we propose a novel sparse vector based encoding approach, which is called Approximate Sparse Coding (ASC). The approach is extended from SC, which has shown excellent performance in image classification [6], [17], but has not been adopted to the large scale human action recognition task. The basic idea of ASC is to compute the sparse codes for a large number of prototype descriptors in the off-line learning phase, and simplify the on-line sparse coding problem to an approximate nearest neighbour searching problem. With such an approximation, the computation cost could be significantly reduced to be acceptable for encoding video features. We extensively evaluated the proposed approach on the KTH dataset and the HMDB51 dataset. We confirmed that ASC is able to encode large number of video features into discriminative low dimensional representations efficiently.

2. Related Works

As mentioned previously, existing local feature encoding approaches can be coarsely divided into two categories according to if the encoding results are sparse vectors or super vectors. Approaches in both categories require a predefined codebook, which could be either a list of prototype local features that are learned using k-means clustering, or a Gaussian Mixture Model (GMM) that is learned to reflect the distribution of local features using the Expectation-Maximization (EM) algorithm. In [16], X. Peng et al. discussed how different codebooks affect the performance of encoding approaches in practice.

Given a codebook of size k , sparse vector based encoding approaches encode a local feature into a k dimensional coefficient vector over the codebook indices. For example, the well-known Vector Quantization (VQ) [4] approach outputs a k dimensional code. The code has one and only one non-zero dimension, which corresponds to the nearest codeword in the codebook, subject to a predefined distance metric (Euclidean distance is usually used as the metric). In VQ, each local feature is assumed to belong to only one codeword. Such a hard assignment causes a lot of information loss and makes the encoding results sensitive to the codebook generation process. For these reasons, Soft Assignment (SA) [18] and Localized Soft Assignment (LSA) [19] were proposed to improve the robustness of VQ. The basic idea of SA and LSA is that a local feature could belong to more than one codewords. In SA, each dimension in the code represents the normalized weight of the local feature to the corresponding codeword. In [19], N. Ballas et al. further introduced the locality constraint to SA and proposed LSA, in which only the normalized weights to the nearest n codewords are reflected in the final code. SA and LSA have slightly better encoding performances comparing to VQ, however, they still require the pair-wise distances

between each local feature and all codewords in the codebook to be computed, which is time consuming.

Sparse Coding (SC) [6] is another kind of sparse vector based coding approach. Different with VQ, the code of SC describes how a local feature could be sparsely reconstructed by the codebook items. The code of SC is very informative because it satisfies both the sparseness constraint and the reconstruction constraint. SC has been adopted in many image categorization works, and produced state-of-the-art results on many datasets [6], [17]. However, in practice, SC suffers severely from the high computational cost of solving the sparseness-constrained reconstruction problem. In [7], Locality-constrained Linear Coding (LLC) was introduced to reduce the computational complexity. In LLC, the reconstruction problem over the whole codebook is simplified to the reconstruction problem over n nearest codewords in the codebook. Due to this simplification, LLC significantly reduces the cost of SC. However, because its first step is to find the n nearest codewords in the codebook, its computational cost is higher than that of VQ, SA or LSA.

Recently, super vector based encoding approaches become very popular. Fisher Vector (FV) [9] encoding is a representative one. FV captures the average first and second order differences between a local feature and the centres of a Gaussian Mixture Model that learned to describe the generating process of the local features. It shares the benefits of both the generative approaches and the discriminative approaches. Vector of Locally Aggregated Descriptors [8] (VLAD) is a hard version of FV with only the first order statistics being preserved. Due to the absence of the second order statistics, VLAD codes are half the dimensionality of FV codes under the same setting. This makes it suitable to be used in encoded higher dimensional local features. However, because the dimensionality of VLAD codes propagates with the dimensionality of local features, its dimensionality could easily become very high when encoding recent popular features such like the powerful CNN (Convolutional Neural Network) features in [20].

Toward efficient local feature encoding, we believe that both the low dimensionality and the high computational efficiency are important. Sparse vector based approaches enjoy a very good property of low dimensionality because they only utilize the zero order statistics. However, in order to make the codes informative, large codebooks are usually used. This causes the computational cost become unaffordable when the quantity of local features is large. On the other hand, super vector based approaches usually utilize a small codebook which results in high encoding speed. However, because higher order statistics are embedded, the dimensionality of the codes is usually very high. This makes it difficult to use them with advanced learning algorithms or elegant action models.

In this paper, we propose Approximate Sparse Coding (ASC) for efficient local feature encoding. The purpose of ASC is similar to LLC, which also introduces an extension of SC. In LLC, the sparseness-constrained reconstruction problem is approximated by using a small number of

Fig. 1 From a large collection of local feature descriptors, we first learn a prototype codebook B_p and a basis codebook B_b . We then utilize B_b to compute the sparse codes for B_p , which results in a sparse code codebook B_{sc} . B_p and B_{sc} have one-to-one correspondences in columns.

local neighbouring codewords. Such a strategy makes the encoding process become faster, but sacrifices some effects of the sparseness constraint. On the other hand, in ASC, the whole on-line encoding process is approximated using ANN search. As it will be confirmed in the experiment, this leads to a significant speed-up over SC and LLC. Furthermore, the sparseness constraint in ASC is fully guaranteed. Though ANN search introduces quantization errors, ASC codes are more discriminative comparing to LLC codes. These properties make ASC a good choice for local feature encoding in the human action recognition task.

It should also be noticed that in the machine learning community, some approximation approaches of sparse coding have been proposed recently. In [21], K. Gregor et al. proposed to learn a parametrized non-linear “encoder” function to predict the sparse codes from inputs. Similarly, in [22], N. Morioka et al. proposed to utilize a piece-wise linear mapping function and a post-refinement procedure for the same purpose. From a structured sparsity perspective, A. Szlam et al. proposed to estimate sparse code by using a tree to assign an input to a group of allowed codebook elements first, and then finding the corresponding coefficient values using a cached pseudoinverse [23]. The common idea of the above approaches is to use parametric models to establish mappings between inputs and their sparse codes. In order to implement them, it is necessary to define the models and learn the parameters carefully in the first place. Different with these approaches, the ANN based approximation approach in our ASC algorithm is a data-driven approach. It has only one more parameter (the size of the prototype codebook) comparing to the traditional sparse coding. This makes it easy to be implemented in practice.

3. Approximate Sparse Coding

The proposed approach is a combination of Sparse Coding (SC) [6] and Approximate Nearest Neighbour (ANN) [24] search. It computes the sparse codes for a large collection of prototype local feature descriptors in the off-line learning phase; and look up the nearest prototype’s sparse code for each to-be-encoded descriptor in the on-line encoding phase. The basic idea underlying ASC is: if the collection of prototypes is large enough to cover sufficient variances in the feature space of local feature descriptors, then encod-

ing a new local feature descriptor can be approximated by taking its nearest prototype’s precomputed sparse code.

Specifically, in the off-line learning phase (Fig. 1), we learn a prototype codebook B_p of size k_p and a basis codebook B_b of size k_b , where $k_b \ll k_p$. B_b is then used to compute one sparse code for each codeword in B_p , which results in a sparse code codebook B_{sc} . In the on-line encoding phase, a local feature is encoded by first finding its approximate nearest prototype in B_p with ANN search, and then taking that prototype’s precomputed sparse code in B_{sc} by a simple indexing operation. The details are presented in the following paragraphs.

3.1 Codebook Learning

Let $X = \{x_i\}_{i=1}^n$ denote a collection of n local feature descriptors that are used in the off-line learning phase, where $x_i \in \mathbb{R}^d$. The prototype codebook $B_p = [b_p^1, b_p^2, \dots, b_p^{k_p}] \in \mathbb{R}^{d \times k_p}$ is learned to partition $\mathbb{R}^d$ into k_p clusters, with each cluster being represented by a prototype codeword $b_p \in \mathbb{R}^d$. The parameter k_p controls how fine the feature space is partitioned. The finer the clusters are, the less variance the local feature descriptors within a same cluster have, i.e., when $k_p = n$, the variance within each cluster is zero.

In our work, in order to obtain a fine enough partition over the feature space, we set the k_p to a very large number and learn the prototype codebook B_p from an even larger collection of randomly collected local feature descriptors. The learning is conducted in K-means clustering by minimizing the following energy:

$$J = \sum_{j=1}^{k_p} \sum_{i=1}^n r_{ij} \|x_i - b_p^j\|^2, \quad (1)$$

where $\|\cdot\|$ is the l_2 norm and r_{ij} is a binary indicator. If x_i is assigned to b_p^j then $r_{ij} = 1$; and if x_i is not assigned to b_p^j then $r_{ij} = 0$. With fixed k_p , B_p is learned by alternatively optimizing between its membership and the data-to-cluster assignment r_{ij} . It is NP-hard and we adopt the Lloyd algorithm [25] as the solver. To speed up the convergence on a collection of millions of local feature descriptors, we also adopt the k-means++ algorithm [26] for initialization.

The second codebook $B_b = [b_b^1, b_b^2, \dots, b_b^{k_b}] \in \mathbb{R}^{d \times k_b}$ is an over-complete set ($k_b \gg d$) of basis vectors that are

learned to capture higher-level information of local feature descriptors. It is used to compute a sparse code codebook $B_{sc} = [b_{sc}^1, b_{sc}^2, \dots, b_{sc}^{k_p}] \in \mathbb{R}^{k_b \times k_p}$, which has one-to-one column-wise correspondences with B_p . A global optimal solution could be obtained by jointly learning on B_b and B_{sc} via the standard sparse coding formula [6]:

$$\min \sum_{j=1}^{k_p} \|b_p^j - B_b b_{sc}^j\|^2 + \lambda \|b_{sc}^j\| \quad (2)$$

$$\text{s.t. } \|b_{sc}^j\| \leq 1, \forall j = 1, 2, \dots, k_p \quad (3)$$

where the first term in Eq. (2) enforces a small error of reconstructing b_p^j from B_b with the coefficient b_{sc}^j , which is known as the reconstruction constraint; the l_1 -norm regularization term of b_{sc}^j enforces b_{sc}^j to have a small number of non-zero elements, which is known as the sparseness constraint; unit l_2 -norm constraint in Eq. (3) is used to avoid trivial solutions, i.e., the objective can be decreased by dividing b_{sc}^j and multiplying B_b respectively using a constant factor.

The global solution of the above problem is very difficult to work out, especially when k_p and k_b are both very large numbers. Furthermore, optimizing B_b and B_{sc} simultaneously leads to the risk of getting a over-fitted B_{sc} . (We did a preliminary experiment in which the feature sign algorithm [27] is used to jointly solve B_b and B_{sc} , and we found both the speed and the performance are not acceptable.) Therefore, in this work, we adopt a straightforward learning strategy, which consists of two steps: (1) learn B_b many times using k-means clustering with fixed k_b , and keep the B_b^* that has the minimal energy (the energy is defined as the average Euclidean distance of the data points to their nearest clusters' centroid); (2) construct B_{sc} using B_b^* by resolving Eq. (2). With our method, the learning of B_b and B_{sc} could be done in hours when using the standard setting ($k_p = 80,000$ and $k_b = 4,000$) of our ASC approach.

3.2 Approximate Coding

Let $q \in \mathbb{R}^d$ be a to-be-encoded local feature descriptor, its approximate sparse code is computed by searching its approximate nearest neighbour in B_p and taking the corresponding sparse code in B_{sc} , which can be implemented as easy as the two-lines algorithm below.

Algorithm 1 Encoding q using B_p and B_{sc}

- 1: $i \approx \arg \min \|b_p^i - q\|$
 - 2: $C_q^{asc} = b_{sc}^i$
-

There are many promising approaches for ANN search [24], which are in principle all compatible to be used in the proposed ASC approach. In this paper, our implementation of ANN is based on the k-d tree structure and follows the Best Bin First (BBF) criterion [28]. Such a method is

easy to be implemented, and has a configurable parameter to control the trade-off between the speed and the accuracy.

Specifically, we utilize a k-d tree structure to index the prototypes in B_p . The k-d tree is constructed by recursively divide the prototypes into two subsets. At each iteration, the variance of every dimension is computed, and the dimension with the largest variance is used to make a cut at the median value of all prototypes on that dimension. Once a cut is made, an internal node is created by saving the dimension id and the cutting value. This results in a balanced tree with depth $t = \log_2 k_p$. The leaves of a k-d tree form a complete partition of the data space, with bins are small in high-density regions and large in low-density regions.

To search the approximate nearest neighbour of q , Best Bin First strategy is utilized. First, the tree is traversed to find the bin containing the query point. This operation only requires t scalar comparisons. Then, bins are checked in the order of increasing distance, which is defined to be the minimum distance between q and any point on the bin boundary. By setting a number E_{max} that indicates how many leaf nodes we are willing to examine, the nearest neighbour in the first E_{max} comparisons are returned as the approximate nearest neighbour. In case E_{max} is set to infinite, the search returns the exact nearest neighbour. However, since E_{max} indicates the times of distance comparison, it directly affects the computational cost.

3.3 Pooling and Normalization

The approximated encoding process that mentioned above only deals with a single local feature descriptor. In practice, in order to obtain the vector representation of a video, it is necessary to compute the codes for all local feature descriptors in that video, and summarize the codes together. This is done through pooling and normalization [29].

In SC and its extension LLC, because the local features are encoded into exact codes, max-pooling is utilized to get the video level vector representation. The resulted vectors capture the maximum activations of the salient characteristics within videos. However, in ASC, due to the quantization errors in the ANN search, the codes are approximated ones. This makes the maximum activations become less accurate. Therefore, we adopt sum-pooling to collect the accumulative activations. Assume $C = \{c_i\}_{i=1}^m$ is the set of m codes computed from an input video, sum-pooling is defined as:

$$p(k) = \sum_{i=1}^m c_i(k), \quad (4)$$

where k denotes the k th component in the code, p is the pooled code; the dimensionality of both p and c is k_b .

In practice, the number of local features m usually varies for different videos. In order to make the pooled code p become less sensitive to m , we further conduct normalization on p . Because in this work, ASC encoding is used with linear SVMs, therefore the optimal normalization is the l_2 -norm normalization according to [30]. Additionally, similar

Fig. 2 The first row shows frames of the six action categories in the KTH dataset (from left to right): boxing, hand clapping, jogging, running, walking, hand waving. The second and the third rows show some sample frames of the action categories in the HMDB51 dataset (from top left to bottom right): brush hair, cartwheel, clap, drink, golf, ride bike, ride horse, run, shake hands, shoot bow, walk, wave.

to Fisher vectors, ASC codes are also sparse vectors. Since dot-product and l_2 distance are poor measures of similarity on sparse vectors, we follow [9] to “unsparisify” the codes by additionally applying power normalization. The power and l_2 -norm normalization we used in ASC is defined as:

$$p_{pw}(k) = \text{sign}(p(k))|p(k)|^{\frac{1}{2}}, \quad (5)$$

$$p_{pwl2}(k) = \frac{p_{pw}(k)}{\|p_{pw}\|}. \quad (6)$$

The resulted p_{pwl2} is the final video-level vector representation. It is ready to be used in learning and recognition.

4. Experiment

We evaluated ASC on two popular action recognition datasets to verify its performance. We also did an extensive study on different configurable settings of ASC. In this section, we present the details.

4.1 Experiment Setting

The experiment were conducted on the KTH dataset and the HMDB51 dataset (some sample frames of these two dataset are shown in Fig. 2). KTH dataset [31] is a relatively small dataset which has only six human action categories including: walking, jogging, running, boxing, waving and clapping. Each action category is performed several times by 25 people. The video sequences were recorded under four different scenarios: indoor, outdoor, outdoor with scale variation, and outdoor with different clothes. In most sequences, the background is static and homogeneous. The dataset consists of 2,391 samples. According to the the original experimental setup in [31], we divide the samples into a training set of 1,528 samples and a testing set of 863 samples. We

follow [31], and report the average accuracy over all categories as the performance measure.

HMDB51 dataset [15] is a large scale dataset which consists of 6,766 trimmed videos labelled in 51 predefined action categories. Most of the videos are taken from movies, and a small number of them are collected from YouTube and other public databases. It is a very challenging benchmark dataset due to its high intra-class variation and low video quality. We follow the original experiment protocol in [15], and used the first train-test split. For each action category, 70 videos are used for training and 30 videos are used for testing, which results in a training set of 3,570 videos and a testing set of 1,530 videos. We report the average accuracy over the 51 categories as the performance measure.

The local feature we adopted in the experiment is the Improved Dense Trajectory (IDT) feature [3] (without human detection). From each video clip, trajectories are densely extracted using the approach and setting described in [3]. Each trajectory is computed into four kinds of descriptors: 30 dimensional Trajectory (TRJ) descriptor, 96 dimensional Histogram of Gradient (HOG) descriptor, 108 dimensional Histogram of Optical Flow (HOF) descriptor and 192 dimensional Motion Boundary Histogram (MBH) descriptor. These descriptors are concatenated to a 426 dimensional raw descriptor for each trajectory.

We implemented five existing approaches, including VQ [4], SC [6], LLC [7], VLAD [8] and FV [9]. As suggested in [11], when implementing these approaches, we adopt PCA to reduce the dimensionality of the raw descriptors to 213 dimensional. We did not further implement Whitening as in [11] because we found the Whitening operation dose not always contribute to the recognition accuracy in our own experiments. In the proposed ASC approach, PCA can be implemented in different timings. By default, it is implemented only in the learning phase, after the B_p

is learned and before learning B_b . In the encoding phase, the raw 426 descriptors are directly used in the ANN search to obtain the approximate sparse code. The impact of PCA timing will be further discussed in Sect. 4.3.

For the five baseline approaches, we trained two codebooks using a set of randomly sampled 512,000 descriptors. One codebook is a Gaussian Mixture Model with 256 components, it is used in VLAD and FV. The other codebook is a list of cluster centres that learned using K-means clustering by setting $k = 4,000$. It is used in VQ, SC and LLC. For ASC, B_p is learned using a set of randomly sampled 5,000,000 descriptors with a fixed k_p , while B_b and B_{sc} are then learned from B_p with a fixed k_b . By default, k_p is set to 80,000; k_p is set to 4,000 for a low dimensional video-level representation ASC-L, or set to 20,000 for a higher dimensional video-level representation ASC-H. The parameter E_{max} , which affect the accuracy of ANN search and the total computational cost, is set to 50. The affect of k_p , k_b , as well as E_{max} will be further examined in Sect. 4.3.

The pooling and normalization methods for the baseline approaches are selected according to the current literature [11]: sum pooling and l_1 -norm normalization are used for VQ; max pooling and l_2 -norm normalization are used for SC and LLC; sum pooling with power and l_2 -norm normalization are used for VLAD and FV. For the proposed approach, the default setting is sum pooling with power and l_2 -norm normalization. In Sect. 4.3, we will also discuss how other settings could affect the results.

The classification is conducted using linear SVM [32] with C setting to 100 as in many recent works [3], [11]. For multi-class classification, we take a one-against-the-rest approach and select the class with the highest score.

4.2 Results and Comparisons

In Table 1, we summarized the average accuracy of the baseline approaches and two configurations of the proposed approach on the KTH dataset. We can confirm that all approaches achieve good recognition accuracy due to the small scale and low difficulty of the dataset. The differences between these approaches are not significant. However, we can still confirm that the super vector based approaches outperform the sparse vector based approaches. Within sparse vector based approaches, when using codebooks of the same size, SC works the best. ASC achieves a good approximation to SC, and outperforms VQ as well as LLC. By comparing between two ASC configurations, it can be confirmed that ASC-H outperforms ASC-L 1.16%, which leads to an explanation that larger codebook leads to better recognition accuracy in the proposed ASC.

We also summarized the average accuracy of these approaches on the HMDB51 dataset in Table 2. On this dataset, the difference of performance between approaches becomes significant. However, the results still show quite similar trends: the proposed ASC achieves a good approximation to SC, and outperforms LLC and VQ. The high computational cost of SC becomes an obvious problem in this

Table 1 Average accuracy on the KTH dataset.

Method	Dimensionality	Average Accuracy
VLAD	54,528	95.13%
FV	109,056	95.83%
VQ	4,000	90.61%
SC	4,000	94.67%
LLC	4,000	91.77%
ASC-L	4,000	91.89%
ASC-H	20,000	93.05%

Table 2 Average accuracy on the HMDB51 dataset.

Method	Dimensionality	Average Accuracy
VLAD	54,528	51.57%
FV	109,056	53.79%
VQ	4,000	36.21%
SC	4,000	46.60%
LLC	4,000	41.31%
ASC-L	4,000	42.22%
ASC-H	20,000	45.82%

Table 3 Encoding costs on 100k raw descriptors.

Method	Time (second)
VLAD	4.36
FV	1.57
VQ	37.77
SC	1433.13
LLC	42.40
ASC-L	2.39
ASC-H	6.48

experiment. The evaluation cost 15 days, which convinces us that SC is not a practical encoding approach for encoding local features of videos. On the other hand, comparing to the super vector based approaches such like VLAD and FV, ASC still performs poorer. However, ASC enjoys a very good property of low dimensionality (the dimensionality of ASC-L codes is 3.67% of the dimensionality of FV codes). This makes ASC be able to be integrated with advanced learning algorithms or elegant action models.

We also evaluated the computational cost of the approaches on a set of randomly sampled 100,000 descriptors. All approaches were implemented in Matlab without taking any care of the speed. The implementations of VLAD, VQ, LLC, ASC-L and ASC-H share a same distance computation function, which guarantees the fairness in comparison. FV is computed based on posterior probability estimation. SC is computed by solving l_1 -norm regularized least squares problems, which is implemented using the feature-sign search algorithm [27] (it is one of the most efficient solver in the literature). The experiment was run on an Intel Xeon E5-1650v3 CPU (3.5GHz) with 64GB RAM. The reported time does not include the computation time of the descriptors, as well as the time used in pooling and normalization. The experiment was run five times, and the average time is reported.

From the results in Table 3, we can confirm that ASC achieves a significant advantage in the computational effi-

Fig. 3 Evaluation of the different aspects of ASC on the KTH dataset and the HMDB51 dataset. Left column: impact of PCA timing; Middle column: impact of the k_p ; Right column: impact of the pooling and normalization methods

ciency comparing to other sparse vector based approaches. ASC-L achieves 600 times speed up comparing to SC, 18 times speed up comparing to LLC, and 16 times speed up comparing to VQ. Comparing to super vector based approaches, ASC-L is faster than VLAD but slightly slower than FV, while ASC-H is slower than both VLAD and FV. It can be confirmed that though ASC uses very large codebooks (more than one order larger than the codebooks used in VLAD and FV), it is still comparable to FV and VLAD in speed. On the other hand, ASC-H is relatively slower than ASC-L. However, it is still significantly faster than other sparse vector based approaches. Considering that ASC-H leads to a 1.16% improvement on the KTH dataset and a 3.6% improvement on the HMDB51 dataset over ASC-L, it is still worth to use it in practice.

4.3 Further Discussions

Similar with many existing encoding approaches, ASC also have some aspects that are configurable. In this section, we conduct a joint evaluation on these aspects using both the KTH dataset and the HMDB51 dataset. The experiment setting follows the descriptions in Sect. 4.1.

We start with a brief description of these configurations: (1) PCA timing: PCA can be adopted right after the descriptor is extracted (dtPCA), or after the ANN search is done (scPCA). In the first case, results of the ANN search can be affected, while in the second case, only the sparse code codebook B_{sc} can be affected. We evaluated these two settings, and one additional setting in which PCA is not

adopted (noPCA); (2) Size of the prototype codebook k_p : k_p affects the ANN search performance. We evaluated five settings for k_p , including 20k, 40k, 60k, 80k and 100k; (3) Size of the basis codebook k_b : k_b affects the quality of B_{sc} . We evaluated five settings of k_b , including 4k, 8k, 12k, 16k and 20k; (4) Pooling and normalization: they are used to obtain the video-level vector representation. For pooling, we evaluated max pooling (MP) and sum pooling (SP); for normalization, we evaluated l_2 -norm normalization (LN), power and l_2 -norm normalization (PL), as well as no normalization (NN).

We first fixed the k_p to 80k and the normalization method to PL, to see how the PCA timing affects the overall performance. The results are plotted in Fig. 3 left. On the KTH dataset, dtPCA works the best, which follows known practices [11]. However, on the HMDB51 dataset, noPCA outperforms dtPCA and scPCA in most settings, which shows a different trend comparing to known practices [11]. We think this is because PCA implements dimension reduction to make raw features become less redundant, but ANN and SC require complete information to get accurate matches and good reconstructions. For small datasets such like KTH, the features are redundant given the small number of categories and training samples, therefore PCA leads to grains in performance. However, for large datasets such like HMDB51, because of the large number of categories and training samples, the features are relatively less redundant. In such a case, PCA sacrifices the quality of ANN search and SC reconstruction, therefore leads to performance loss.

Fig. 4 Evaluation results on the affects of E_{max} .

We then fixed the PCA timing to scPCA, the normalization approach to PL and the the pooling approach to SP, to see how the changes in k_p affect the action recognition performance. The results are plotted in Fig. 3 middle. Generally, a large k_p leads to a better overall performance. However, the differences between different k_p are not obvious on both datasets. This is to say, it is not a good idea to try to improve ASC by using a larger k_p . Different with k_p , on the HMDB51 dataset, it is obvious that k_b has a trend that the larger it is, the better the performance becomes. Though this trend is not obvious on the KTH dataset, we believe this is simply because the KTH dataset is small and easy that even a small k_b can lead to good enough performance. Such results suggest us that it is worth to use large k_b when working on large and difficult datasets, even it may lead to additional computational costs.

In the last, we fixed the PCA timing to scPCA and k_p to 80k, to see how different pooling and normalization strategies affect the performance. The results are plotted in Fig. 3 right. Comparing to the impacts of PCA timing and k_p , normalization and pooling strategy are obviously more significant for the proposed ASC. On both datasets, it is quit clear that PL works as the best normalization approach. It produces the best results on both datasets when combined with SP. The curve also suggests that as k_b getting large, the performance of max pooling becomes close to the performance of sum pooling.

Beside the configurations that examined above, there is another internal parameter E_{max} in the ANN search (Sect. 3.2). It affects the trade-off between accuracy and speed in the ANN search. We evaluate the affect of E_{max} by fixing the k_p to 80k, the k_b to 4k, the PCA timing to scPCA, the pooling method to SP, the normalization method to PL; and evaluate the recognition accuracy by varying on E_{max} . We report the average accuracy over both the KTH dataset and the HMDB51 dataset, and the 5-round-average computation cost on 100,000 random samples local descriptors. The results are plotted in Fig. 4.

It can be confirmed that the computational cost propagates with E_{max} . However, the recognition performance does not change significantly on both datasets. This suggests us that it is no harm to use a small E_{max} in ASC. In

our opinion, ASC captures the approximate sparseness information of local features. As long as the encoding is done under a same process, the quantization errors in ANN does not affect too much in practice.

5. Conclusion

In this paper, we proposed Approximate Sparse Coding for the human action recognition task. We evaluated ASC on popular benchmark datasets, and confirmed it has the ability to encode large quantity of local video features into discriminative low dimensional vector representations efficiently.

In the future work, we will make full use of the low dimensionality of ASC code to develop novel spatial temporal pyramid based video representations. Beside, we will also implemented ASC with other ANN search algorithms to further boost the speed.

Acknowledgements

This work is supported by the JSPS Grant-in-Aid for Scientific Research B (No.26280057) and the JSPS Grant-in-Aid for Challenging Exploratory Research (No.26540081). Yu Wang is also supported by a JSPS Postdoctoral Fellowship for Foreign Researchers (No.2604043).

References

- [1] I. Laptev and T. Lindeberg, "Space-time interest points," Proceedings of International Conference on Computer Vision, vol.1, pp.432–439, 2003.
- [2] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.3169–3176, 2011.
- [3] H. Wang and C. Schmid, "Action recognition with improved trajectories," Proceedings of International Conference on Computer Vision, pp.3551–3558, 2013.
- [4] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," Proceedings of International Conference on Computer Vision, vol.2, pp.1470–1477, 2003.
- [5] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," Proceedings of British Machine Vision Conference, pp.124.1–124.11, 2009.
- [6] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.1794–1801, 2009.
- [7] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.3360–3367, 2010.
- [8] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.3304–3311, 2010.
- [9] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," Proceedings of European Conference on Computer Vision, vol.6314, pp.143–156, 2010.
- [10] X. Zhou, K. Yu, T. Zhang, and T. Huang, "Image classification using super-vector coding of local image descriptors," Proceedings of European Conference on Computer Vision, vol.6315, pp.141–154,

- 2010.
- [11] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," arXiv:1405.4506.
 - [12] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors," *Proceedings of European Conference on Computer Vision*, vol.8693, pp.581–595, 2014.
 - [13] S. Lazebnik, C. Schmid, and J. Ponce., "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp.2169–2178, 2006.
 - [14] N. Ballas, Y. Yang, Z.-Z. Lan, B. Delezoide, F. Preteux, and A. Hauptmann, "Space-time robust video representation for action recognition," *Proceedings of International Conference on Computer Vision*, pp.2704–2711, 2013.
 - [15] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition," *Proceedings of International Conference on Computer Vision*, pp.2556–2563, 2011.
 - [16] X. Peng, L. Wang, Y. Qiao, and Q. Peng, "A joint evaluation of dictionary learning and feature encoding for action recognition," *Proceedings of International Conference on Pattern Recognition*, pp.2607–2612, 2014.
 - [17] T. Ge, Q. Ke, and J. Sun, "Sparse-coded features for image retrieval," *Proceedings of British Machine Vision Conference*, pp.132.1–132.11, 2013.
 - [18] N. Ballas, Y. Yang, Z. Lan, B. Delezoide, F. Preteux, and A. Hauptmann, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.32, pp.1271–1283, 2009.
 - [19] N. Ballas, Y. Yang, Z. Lan, B. Delezoide, F. Preteux, and A. Hauptmann, "In defense of soft-assignment coding," *Proceedings of International Conference on Computer Vision*, pp.2486–2493, 2011.
 - [20] Z. Xu, Y. Yang, and A. Hauptmann, "A discriminative cnn video representation for event detection," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp.1798–1807, 2015.
 - [21] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," *Proceedings of International Conference on Machine Learning* 2010.
 - [22] N. Morioka and S. Satoh, "Generalized lasso based approximation of sparse coding for visual recognition," *Proceedings of The Twenty-fifth Annual Conference on Neural Information Processing Systems*, 2011.
 - [23] A. Szlam, K. Gregor, and Y. LeCun, "Fast Approximations to Structured sparse coding and Applications to Object Classification," *Proceedings of European Conference on Computer Vision*, pp.200–213, 2012.
 - [24] M. Muja and D.G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.36, no.11, pp.2227–2240, 2014.
 - [25] S. Lloyd, "Least square quantization in pcm," *IEEE Transactions on Information Theory*, vol.28, no.2 pp.129–137, 1982.
 - [26] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," *Proceedings of the 18th Annual ACM-SIAM symposium on Discrete Algorithms*, 2007.
 - [27] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Proceedings of Annual Conference on Neural Information Processing Systems*, 2007.
 - [28] J. Beis and D.G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp.1000–1006, 1997.
 - [29] Y.L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," *Proceedings of International Conference on Machine Learning* 2010.
 - [30] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp.3539–3546, 2010.
 - [31] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach," *Proceedings of International Conference on Pattern Recognition*, vol.3, pp.32–36, 2004.
 - [32] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol.9, pp.1871–1874, 2008.

Yu Wang received the M.S. degree in Information Science and Ph.D. degree in Engineering, from Nagoya University, in 2010 and 2013, respectively. He is currently a postdoctoral researcher with the Graduate School of Information Science, Nagoya University. His research interests are object recognition and action categorization. He is a member of IEEE.

Jien Kato received the M.E. and Ph.D. degrees in Information Engineering from Nagoya University in 1990 and 1993, respectively. She is currently an Associate Professor with the Graduate School of Information Science, Nagoya University. Her research interests include computer vision, machine learning, multi-sensor perceptual computing and their applications. She is a member of IEICE, IPSJ, and a senior member of IEEE.