Towards Good Practices for Action Video Encoding

Jianxin Wu
National Key Laboratory for Novel Software Technology
Nanjing University, China
wujx2001@nju.edu.cn

Yu Zhang
Nanyang Technological University
Singapore
roykimbly@hotmail.com

Weiyao Lin
Shanghai Jiao Tong University
China
wylin@sjtu.edu.cn

Abstract

High dimensional representations such as VLAD or FV have shown excellent accuracy in action recognition. This paper shows that a proper encoding built upon VLAD can achieve further accuracy boost with only negligible computational cost. We empirically evaluated various VLAD improvement technologies to determine good practices in VLAD-based video encoding. Furthermore, we propose an interpretation that VLAD is a maximum entropy linear feature learning process. Combining this new perspective with observed VLAD data distribution properties, we propose a simple, lightweight, but powerful bimodal encoding method. Evaluated on 3 benchmark action recognition datasets (UCF101, HMDB51 and Youtube), the bimodal encoding improves VLAD by large margins in action recognition.

1. Introduction

The past decade has witnessed increasing interests on action recognition in videos (*e.g.*, [10, 21, 20, 12, 9, 24]). Many methods in this vast literature are based on the bag-of-features video encoding scheme. Without loss of generality, we can decompose a bag-of-features based action recognition pipeline into three steps:

- **Raw motion feature extraction**. We want to encode motion, arguably the most important cues for action recognition. Examples of popular raw motion feature include the space-time interest points (STIP) [10] and dense trajectory features (DTF) with motion boundary histograms (MBH) [24].
- **Action video encoding**. After raw motion features are extracted from a video, bag-of-features encodes this set into a single vector (*e.g.*, [21, 11, 9, 24]).
- **Learning**. Various classifiers (*e.g.*, support vector machines) learn action recognition models.

Among these steps, motion features have enjoyed the most attentions in the past. The DTF features in [24] significantly outperformed previous state-of-the-art results on 9 benchmark datasets. As one example, recognition accuracy on the challenging HMDB51 dataset jumped from 26.9% (in [17]) to 48.3% (in [24]). Finding good motion features may continue to maintain its central role in action recognition research, e.g., the ω-flow features further increased accuracy on HMDB51 to 52.1% [6].

The VLAD encoding framework was used in [6] instead of the classic bag-of-features, a fact that contributed a lot to its accuracy gain. One very recent trend is that high-dimensional encoding frameworks such as Fisher Vector (FV) [18] or VLAD [7] are gradually gaining popularity in video representation—mainly due to their excellent recognition accuracies [25]. For example, VLAD exhibited significant improvements on two difficult datasets (Hollywood2 and HMDB51) over bag-of-features [6]. In [23], using FV or VLAD can both improve the event classification accuracy by a large margin.

In this paper, through empirical evaluations, we show that *proper video encoding can further boost action recognition accuracy, and the gains are achieved with negligible increase of computational cost*. Specifically,

- **Empirical VLAD evaluation.** Many improvements have been proposed for FV/VLAD image encoding, including at least power normalization [16], root descriptors [1], spatio-temporal pyramid (STP) [11] (an extension of SPM to videos, see also [24, 23]) and residue normalization [3]. We evaluated these techniques and show that: except for power normalization and STP, they not always improve action recognition rates (different from previously reported image classification / retrieval results). Note that the aspects evaluated in this paper are complementary to those evaluated in [25, 14].
- **VLAD is maxent feature learning.** We propose a novel interpretation for VLAD: as a maxent (maximum-entropy) feature learning process. Combining with empirical properties of generated VLAD vectors, this new perspective provides explanations for the success and deficiency of the techniques evaluated above. We then provide practical suggestions on when to use (or not to

use) these techniques.

- **Bimodal encoding.** Based on the maxent interpretation, we propose bimodal encoding. Empirical evaluations show that this simple transformation buys us good accuracy gains with almost no computational overhead.

Overall, the goal of this paper is to provide a set of good practices for action video encoding. When we are given a type of raw motion feature (or set of motion features), good video encoding can make the best of its representational power. And, when we are comparing different motion features, proper video encodings can remove the bias of video encodings to ensure fair comparisons between features.

We start by introducing video encoding techniques and their evaluations in Sec. 2, followed by the maxent interpretation and observations of VLAD vectors in Sec. 3. The proposed encodings, evaluations, and comparisons are presented in Sec. 4. Sec. 5 concludes this paper.

2. Background and VLAD Evaluation

Given an action video, suppose a set of N motion features are extracted from it and a codebook of size K is learned by the k-means clustering algorithm, we denote the code words in the codebook as $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$ and the motion features as $\boldsymbol{x}_i^j$, where $i = 1, \ldots, K$ and $j = 1, \ldots, n_i$. In this notation, $\boldsymbol{x}_i^1, \ldots, \boldsymbol{x}_i^{n_i}$ are all the motion features that belong to the i-th code word (*i.e.*, whose nearest neighbor in the codebook is $\boldsymbol{c}_i$). We assume that $\boldsymbol{c}_i, \boldsymbol{x}_i^j \in \mathbb{R}^d$ for all i and j, and $\sum_{i=1}^K n_i = N$.

The classic bag-of-features encoding will encode this video as a vector in $\mathbb{R}^K$, as

$$(n_1, n_2, \cdots, n_K)^T \in \mathbb{R}^K; \tag{1}$$

while the VLAD encoding is

$$\boldsymbol{v} = \left(\sum_{j=1}^{n_1} (\boldsymbol{x}_1^j - \boldsymbol{c}_1), \ldots, \sum_{j=1}^{n_K} (\boldsymbol{x}_K^j - \boldsymbol{c}_K) \right) \in \mathbb{R}^{d \times K}. \tag{2}$$

VLAD vectors are usually ℓ_2 normalized by $\boldsymbol{v} \leftarrow \boldsymbol{v}/\|\boldsymbol{v}\|$.

There are several improvements to the VLAD encoding in the image classification or retrieval community:

- Power normalization [16]. Apply $v \leftarrow \mathrm{sgn}(v)|v|^\alpha$, $0 \le \alpha \le 1$ after Eq. 2, but before ℓ_2 normalization.
- Root descriptors [1]. Apply $\boldsymbol{x}_i^j \leftarrow \sqrt{\boldsymbol{x}_i^j}$ before Eq. 2, assuming $\boldsymbol{x}_i^j$ is a non-negative vector for all i and j.
- Residue normalization [3]. Replace $\boldsymbol{x}_i^j - \boldsymbol{c}_i$ by $\frac{\boldsymbol{x}_i^j - \boldsymbol{c}_i}{\|\boldsymbol{x}_i^j - \boldsymbol{c}_i\|}$ in Eq. 2 for all i.
- Spatio-temporal pyramid (STP) [11]. We use the following setup: split the video horizontally into 2 halves, apply Eq. 2 to extract VLAD from the both the original video and these two halves, leading to a $3 \times\ d \times K$ dimensional video encoding.

Note that in VLAD $\boldsymbol{x}_i^j$ are *not* raw motion features. They are after PCA dimension reduction because PCA is essential for the success of FV and VLAD [18, 7].

We evaluate the performance of these techniques using three benchmark datasets. They cover different scales and difficulty levels for action recognition:

UCF101 [22]: A large scale dataset with 101 action categories and 13320 clips. The protocol of [8] is used: run on 3 predefined splits and report the average accuracy.

HMDB51 [9]: A medium scale dataset with 51 actions in 6766 clips. We use the original (not stabilized) videos and follow [9] to report average accuracy of its 3 splits.

Youtube [12]: A small scale dataset with 11 action types. Following the original protocol, we report 25-fold cross validation accuracy rates.

Because our focus is video encoding, we fix the raw motion feature to the dense trajectory features (DTF) [24] and the classifier to linear SVM. In DTF, we use all its five feature types (trajectory, HOG, HOF, mbhX and mbhY). We first sample roughly 10^6 raw motion features from 10% videos in each dataset, then perform PCA to keep 90% energy. The PCA operations are performed separately for each of the 5 feature types. We use default parameters of DTF which results in 426 dimensions in total. After PCA, the motion features reduce to approximately 200 dimensions in different runs of all 3 datasets. We use LIBLINEAR [4] with default parameters for classification. When performing power normalization, we set $\alpha = 0.5$.

Evaluation results are summarized in Table 1, in which K stands for the codebook size, PM for power normalization, RD for root descriptors, RN for residue normalization, and STP for spatio-temporal pyramid. We not only list the average accuracy in different runs, but also individual run's accuracy rate. The following observations can be drawn from this table:

- Both power normalization and increasing K are always useful. Comparing within blocks 1–3 and within 4–6 clearly shows that when K increases, accuracy consistently increases in every run (and of course leads to higher averages). It is also obvious that increasing K is more effective in difficult datasets (HMDB51). Its effect in UCF101 is smaller than that in HMDB51 and even diminishes in the easy Youtube dataset (*c.f.* blocks 4–6). Power normalization seems equally effective. Comparing block 1 vs. 4, 2 vs. 5 and 3 vs. 6, power normalization brings around 2.5% accuracy gains, except when $K = 256$ (but still with 1% higher accuracy).
- Root descriptors and residue normalization are sometimes useful. Comparing block 7 to block 6, root descriptors slightly improves accuracy in UCF101, is a little bit worse in HMDB51, but incurs 3% loss in Youtube. Since the changes are small in 2 datasets, we performed additional experiments (*e.g.*, with STP) and

Table 1. Comparison of VLAD improvement techniques. '#' is a block number in the table, which helps refer to different experiments. 'Ave' is the average of train/test split 1, 2, and 3 in UCF101 and HMDB51, or the cross-validation accuracy in Youtube.

#	Run	K	PM	RD	RN	STP	UCF	HMDB	Youtube
1	1	64					72.88%	42.75%	
	2	64					74.58%	42.29%	
	3	64					75.60%	44.31%	
	Ave	64					74.35%	43.12%	84.09%
2	1	128					75.13%	46.14%	
	2	128					76.97%	46.27%	
	3	128					77.73%	46.21%	
	Ave	128					76.61%	46.21%	84.64%
3	1	256					77.06%	48.63%	
	2	256					77.34%	49.02%	
	3	256					78.60%	49.35%	
	Ave	256					77.67%	49.00%	85.45%
4	1	64	✓				75.76%	45.62%	
	2	64	✓				76.81%	44.84%	
	3	64	✓				78.08%	46.67%	
	Ave	64	✓				76.88%	45.71%	88.09%
5	1	128	✓				77.77%	48.82%	
	2	128	✓				78.82%	48.10%	
	3	128	✓				80.33%	48.95%	
	Ave	128	✓				78.97%	48.63%	88.09%
6	1	256	✓				79.49%	49.54%	
	2	256	✓				80.18%	49.67%	
	3	256	✓				80.52%	51.18%	
	Ave	256	✓				80.06%	50.13%	88.09%
7	1	256	✓	✓			79.41%	49.41%	
	2	256	✓	✓			80.45%	50.00%	
	3	256	✓	✓			80.65%	50.71%	
	Ave	256	✓	✓			80.17%	50.04%	85.00%
8	1	256	✓		✓		78.98%	49.41%	
	2	256	✓		✓		79.99%	49.54%	
	3	256	✓		✓		81.09%	51.05%	
	Ave	256	✓		✓		80.02%	50.00%	86.82%
9	1	256	✓			✓	83.19%	53.92%	
	2	256	✓			✓	83.88%	54.77%	
	3	256	✓			✓	84.71%	55.10%	
	Ave	256	✓			✓	83.93%	54.60%	89.82%

observed the same trend: better for UCF101, but worse for others. Residue normalization, however, leads to lower accuracy rates in all 3 datasets.

- STP is useful in all 3 datasets. Comparing blocks 9 with block 6, STP improves the accuracy by 3.7%, 4.5% and 1.8%, respectively.

We make the following suggestions of good practices based on evidences from Table 1:

- *Complexity is an important factor.* Techniques that improve performance by introducing longer features (*i.e.*, increasing K and using STP) is usually effective, but the scale of improvement is related to difficulty of the problem at hand. Thus, it is good to start with smaller K, and gradually increase the codebook size or the number of divisions in STP, until the performance is saturated.
- *Residue normalization likely will not change video encoding quality.* Thus, we will not further experiment with this technique in this paper.

However, important questions remain: 1) Why root descriptors have mixed results and when will it help? 2) Why is power normalization always effective in video encoding? We will handle both questions in the next section.

3. VLAD: Distribution of values & A MaxEnt Interpretation

In this section, we will have a closer look of the VLAD framework to understand root descriptors and power normalization better. Given a raw motion feature $\tilde{\boldsymbol{x}} \in \mathbb{R}^{\tilde{d}}$, we first perform the PCA to get our motion feature $\boldsymbol{x}$

$$\boldsymbol{x} \leftarrow P\left(\tilde{\boldsymbol{x}} - \mathrm{E}(\tilde{\boldsymbol{x}})\right) , \tag{3}$$

where $P \in \mathbb{R}^{d \times \tilde{d}}$ and $\mathrm{E}(\tilde{\boldsymbol{x}})$ are eigenvectors of the covariance matrix of $\tilde{\boldsymbol{x}}$ and its mean, respectively. Further assuming we center the raw motion features (*i.e.*, minus $\mathrm{E}(\tilde{\boldsymbol{x}})$ from them), Eq. 3 reduces to $\boldsymbol{x} \leftarrow P\tilde{\boldsymbol{x}}$.

In order to simplify the notations, in this section we assume $K = 1$ without loss of generality. All the raw features from a video are denoted as $\tilde{\boldsymbol{x}}^j$, $j = 1, \ldots, N$; and after PCA they become $\boldsymbol{x}^j = P\tilde{\boldsymbol{x}}^j$. There is only one code word, which we call $\boldsymbol{c}$.[1] Then, the VLAD vector is

$$\boldsymbol{v} = \textstyle\sum_{j=1}^{N}(\boldsymbol{x}^j - \boldsymbol{c}) = P\left(\sum_{j=1}^{N} \tilde{\boldsymbol{x}}^j\right) - N\boldsymbol{c} . \tag{4}$$

Eq. 4 states that given the input (sum of raw motion features belonging to a given code word), the VLAD vector $\boldsymbol{v}$ contains values of several linear feature functions, whose projection directions are determined by PCA eigenvectors.

In other words, given an input video, VLAD is in effect a two step process: 1) reduce the set of raw features belonging to a code word into a single vector (sum of raw features); and, 2) compute values of linear feature functions.[2]

Linear feature functions have been widely used, *e.g.*, in natural language processing. In the literature, maximum-entropy (maxent) is a widely used method for supervised and unsupervised linear feature function learning [13]. The maxent principle indicates that in an unsupervised case (*i.e.*, without prior knowledge or constraint), we should seek linear projections that lead to maximum entropy of the projected data ($P\sum_j \tilde{\boldsymbol{x}}^j$ in our case).

It is well known that if the input data ($\tilde{\boldsymbol{x}}^j$) follows a normal distribution, then the PCA eigenvectors with largest eigenvalues are the optimal maxent solutions [5].[3] Thus, we arrive at the conclusion that *VLAD can be viewed as a maximum-entropy linear feature learning process, which is optimal when the raw motion features are Gaussian.*

[1] Since $K = 1$, we will drop the subscripts from $\boldsymbol{c}$ and $\boldsymbol{x}$.

[2] Here we do not take into account the effect of ℓ_2 normalization, which can be viewed as a subsequent postprocessing step.

[3] For completeness, we provide a sketch of proof in the Appendix.

3.1. Normality and root descriptors

Since normality is the assumption for optimal maxent solutions and for PCA to produce statistically independent dimensions (which is suitable for linear classifiers), it is thus desirable to have Gaussian raw features. However, normality rarely holds for either image or motion features, as will be shown by the examples in Fig. 1. These facts lead to a conjecture that *the success or deficiency of root descriptors hinges on whether it can improve the normality.*

Although there is no intuitive way to check normality of multivariate distributions (in our case $\sum_j \tilde{\boldsymbol{x}}^j$), we can check normality of a single linear feature function (*i.e.*, one dimension in $P\sum_j \tilde{\boldsymbol{x}}^j$). If normality holds for raw features, the extracted features must also be normal, because linear combination of Gaussians is once again a Gaussian.

The normality of a one-dimensional random variable can be tested intuitively using the normal probability plot [2]. In Fig. 1 we show normal probability plots for all three datasets, with and without power normalization, in the first row inside each subfigure. The plots are using data from a randomly chosen VLAD dimension (dimension 36441 in Fig. 1). A normal probability plot has the data value in the x-axis, and the data percentiles in the y-axis in log-scale. When the empirical distribution (blue '+' signs) fits well to the groundtruth normal probability line (red dashed line), the data follows a normal distribution. It is easy to observe that in all datasets, the VLAD vector dimension (*i.e.*, one linear projection feature) deviates from the red line significantly (Fig. 1a, 1c, and 1e).

The application of root descriptors, however, can sometimes improve normality, as shown in Fig.1b, 1d and 1f. In UCF101, except for the tail (data above the 0.9 percentile), the curve fits well to a normal distribution, that is, the normality is dramatically improved. There is no clear difference for HMDB51 and Youtube. Another intuitive indicator of normality is the histogram of data values, as shown in the top-left plot in every subfigure. Without root descriptors, most distributions have sharper modes than what a normal distribution expects (*e.g.*, having closer shape to a Laplace than to a normal distribution). The root descriptors, however, is similar to a Gaussian in UCF101. These observations all coincide well with the results in Table 1: root descriptors help in UCF101, but in neither HMDB51 nor Youtube.

Fig. 1 only visualizes one dimension in the VLAD vector. We use statistical tests to examine the normality of a large number of dimensions. We group all dimensions in the VLAD vector into groups, with each group containing 128 contiguous dimensions. Then we use the Jarque-Bera test in Matlab to examine the first dimension in each group. This function sets any p-value less than 0.001 to 0.001. Because of the heavy tails and the spikes at zero (*c.f.* Fig. 1), we remove all values that exactly equal 0 and those data points outside of the $[0.01 \quad 0.99]$ percentile range in this test. We count the number of non-trivial p-values (bigger than 0.001) as the indicator—we are not expecting an exact normal distribution, instead, we care about some VLAD dimensions that are close to normal except in the tails.

Table 2. Normality test results: mean and max of Jarque-Bera tests' p-values, and the number of non-trivial p-values. A large p-value indicates that the distribution is likely a normal distribution.

	mean	max	#non-trivial
UCF101	0	0.38	16
UCF101-Root	0.01	0.45	20
HMDB51	0.02	0.5	56
HMDB51-Root	0.02	0.5	47
Youtube	0.04	0.5	112
Youtube-Root	0.04	0.5	103

Table 2 summarizes the normality results. When root descriptors are applied to raw motion features, UCF101 shows more non-trivial p-values and increased mean and maximum p-values than the case where root descriptors are not used. The reverse phenomena are observed in HMDB51 and Youtube. Again, we see that the trends in normality tests (Table 2) coincide well with the trends in accuracy rates (Table 1).

In short, putting these observations together, we have reasons to conjecture that *root descriptors are improving action recognition accuracy through improving normality of the raw motion features—that is, make the normality assumption of maxent feature learning more realistic.*

The normality of root descriptors, however, are still weak. Average p-values in Table 2 are all less than 0.05 (mainly due to the heavy tails); and it does not seem easy to find data transformation that could make motion features more Gaussian-like. Thus, it might be advantageous to directly explore the maxent linear projections to replace the PCA eigenvectors. We leave this topic to future research.

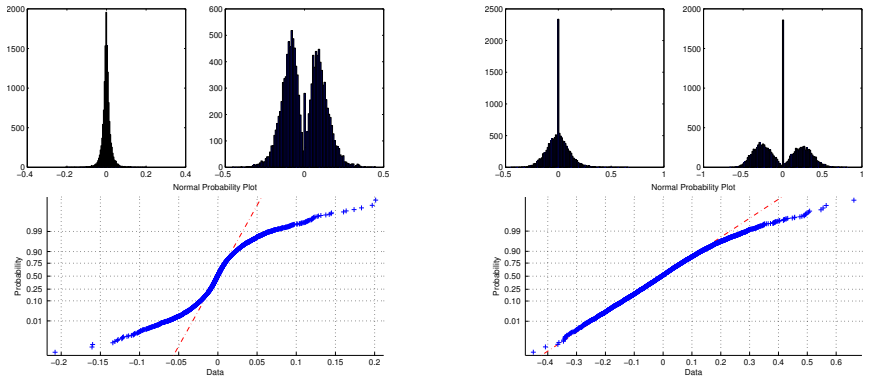
Although we focus on VLAD in this paper, we want to add a note that the maxent analyses also readily apply to Fisher Vectors.

3.2. Side information, power normalization and bimodal distribution

We also visualize histograms after power normalization in the top right corner in every subfigure of Fig. 1. These plots reveal some video VLAD encoding properties that are different from their image counterparts.

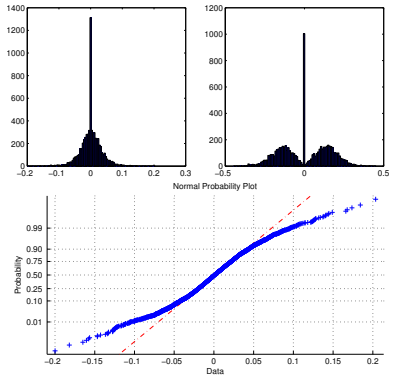
3.2.1 Missing code words as side information

One obvious common pattern in Fig. 1 is the existence of a tall bar at the x-axis' zero point, *i.e.* many values are *exactly* 0. Table 3 shows the percentages of zero entries in VLAD vectors. From Table 3, all datasets have 15% to 20% zero

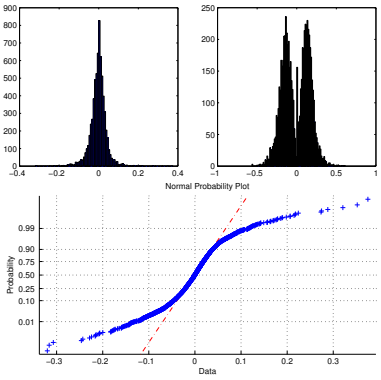


(a) UCF101

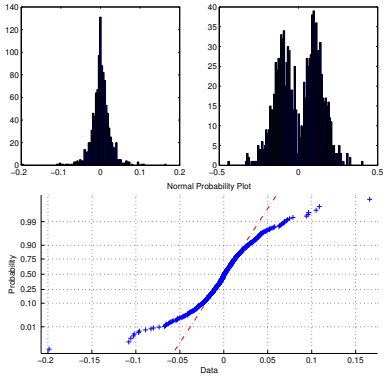
(b) UCF101 (Root descriptors)



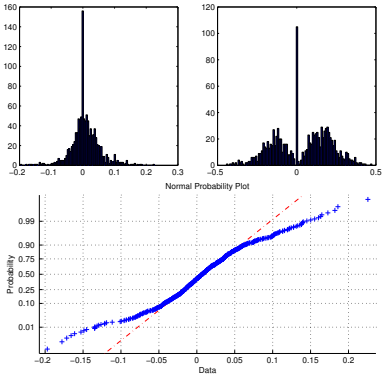
(c) HMDB51



(d) HMDB51 (Root descriptors)



(e) Youtube



(f) Youtube (Root descriptors)

Figure 1. Distribution of one dimension (36441-th) of all videos in a dataset in the VLAD vector: 1a, 1c and 1e are for the UCF101, HMDB51 and Youtube without using root descriptors, respectively; while 1b, 1d and 1f are with the root descriptors technique. In the first row of a subfigure, the left plot shows a histogram without power normalization; while the right plot is after power normalization. The second row shows the normal probability plot [2]. Plots are based on VLAD vectors without the ℓ_2 normalization. (Best if viewed in color.)

Table 3. Percentages of zero entries in VLAD vectors. Datasets are generated using the setup of block 6 and 7 in Table 1, respectively.

UCF	HMDB	Youtube
15.59%	18.44%	14.21%
UCF-RD	HMDB-RD	Youtube-RD
16.95%	20.33%	15.17%

entries.

In Eq. 2, if $n_j > 0$ (*i.e.*, number of raw motion features belonging to a code word is non-zero), it is rarely possible that its corresponding entires in $\boldsymbol{v}$ will be zero. Thus, Table 3 indicates that on average, more than 15% code words are missing in any single video—a fact that carries useful side information for categorizing a video's action type. A common VLAD practice is to encode a missing code word as $\mathbf{0} \in \mathbb{R}^d$. However, this practice completely ignores the useful side information.

3.2.2 Bimodality from power normalization

If we ignore the tall bars at zero, the histograms using power normalization in Fig. 1 clearly show bimodal distributions for all datasets. Although Fig. 1 only visualizes one dimension, we observe the same phenomenon in all 10 other randomly chosen VLAD dimensions. This property makes video encoding different from some image encoding results. In Fig. 1 of [16], distributions before power normalization are sharp and long-tailed, similar to our Fig. 1a, 1c and 1e. However, after power normalization, its Fig. 1d is still unimodal, unlike our Fig. 1b, 1d and 1f.

In the unimodal histograms, almost all values squeeze in a small range around the zero point, meaning that the features have small discriminative power. The bimodal distributions are more discriminative, which explains why the simple power normalization technique has significantly improved action recognition results.

3.2.3 *Scale = Importance?*

One final important fact we observe is that the scales of different dimensions in the VLAD vector are diverse. Fig. 2 shows the histograms of maximum values of all VLAD dimension for the 3 datasets. It is clear that the scales of different dimensions vary significantly.

A dimension with larger scale will (in practice) be more important in classification than a dimension with smaller scales. However, there is no evidence that dimension scale variations are directly related to their importance levels for categorizing actions. For example, we examined the scales of only those dimensions corresponding to the eigenvector with the largest eigenvalue. This subset are considered as containing the most information in PCA, but the resulting histogram has similar shape and range as those shown in Fig. 2. Thus, we believe that in VLAD, $scale \neq importance$, and it is a good practice to remove the scale differences.

In short, based on careful analyses of VLAD's data distribution properties and the maxent interpretation, we suggest the following good practices: 1) use root descriptor if it can improve normality; 2) make good use of the side information (*i.e.*, which code words are missing in a video); 3) take advantage of the bimodal distribution in power normalization; and, 4) remove the scale differences.

4. Bimodal encoding

In this section, we propose a few simple encoding methods that incorporate various practices suggested in Sec. 3, and empirically evaluate their performance. We show that what we name as the *bimodal encoding* is the most effective in experiments.

Table 4. Evaluation results (average accuracy) of different encoding choices, comparison with state-of-the-art results, and feature compression results. We fix $K = 256$ and use STP for all datasets, and use root descriptors only in UCF101. These choices are based on discussions in Sec. 3.

	UCF101	HMDB51	Youtube
none	84.12%	54.60%	89.82%
Even-Scale	83.73%	55.80%	**90.82**%
Side-Info	83.98%	**56.36**%	90.18%
Bimodal	**84.16**%	56.14%	90.36%
2-bit Bimodal	82.41%	53.88%	89.00%
1-bit Bimodal	81.24%	51.90%	88.18%
bag-of-features	78.43%	48.3 % [24]	85.4 % [24]
[19] (FV)	n/a	37.21%	84.52%
[6] (VLAD)	n/a	52.1 %	n/a

4.1. Encoding transformations

These encodings are all per-dimension transformations. Suppose x a variable representing one dimension in VLAD (after power and ℓ_2 normalization), we denote $\overline{x}(> 0)$ and $\underline{x}(< 0)$ as the maximum and minimum value of this dimension observed in the training set, respectively. The encoding transformations are defined as:

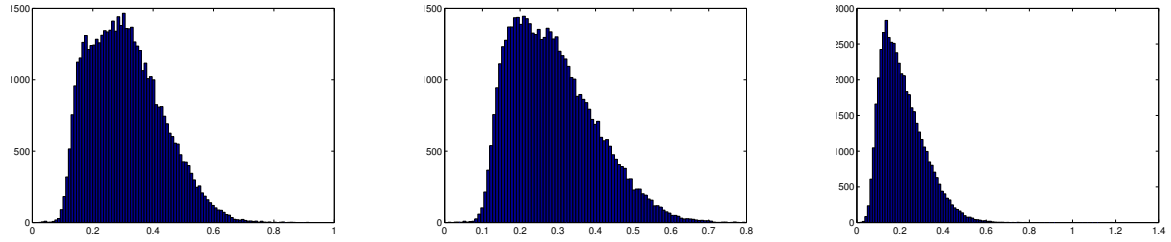
- *Even-Scale*. $x \leftarrow \frac{x}{\overline{x}-\underline{x}}$. The new x has varying minimum and maximum values in each dimension, but the difference between minimum and maximum in each dimension is always 1. Note that 0 remain unchanged.
- *Side-Info*. $x \leftarrow \frac{x-\underline{x}}{\overline{x}-\underline{x}}$. Note that 0 will be transformed to $\frac{-\underline{x}}{\overline{x}-\underline{x}}$, thus it is a simple way to incorporate the side information. The new range is $[0\ 1]$ in all dimensions.
- *Bimodal*. x is transformed as follows: $x \leftarrow x/\overline{x}$ if $x \geq 0$ and $x \leftarrow x/|\underline{x}|$ if $x < 0$. This encoding handles the bimodality explicitly and fix scales in both sides of 0. The new range is $[-1\ 1]$. Note that 0 remain unchanged.

All three transformations are computational efficient. In order to get the parameters $\overline{x}$ and $\underline{x}$, we just need 1 comparison and 1 assignment operation for each feature value x. In order to apply the transformations, only 1 or 2 minus and 1 division operation are needed for each x.

4.2. Comparison of encodings and comparison with state-of-the-art

Evaluation results for different encoding transformations are presented in the first block of Table 4. The first row in Table 4 ("none") are accuracy rates without transformations. Note that in UCF101, using root descriptors increased the STP accuracy from 83.93% (block 9 of Table 1) to 84.16%.

The following 3 rows transform data in the first row's experiments using the proposed transformations. Except in the UCF101 dataset, all transformations can improve accuracy rates, and the bimodal encoding is the most effective. These results confirms our observations: $scale \neq$



(a) UCF101 (b) HMDB51 (c) Youtube

Figure 2. Distribution of maximum values of all VLAD dimensions. Because in every dimension the distribution is almost symmetric (*c.f.* Fig. 1), we do not show histograms of minimum values.

importance, and both bimodal and side information are useful. In the UCF101 dataset, even-scale and side-info slightly reduces the accuracy by 0.39% and 0.14%, respectively; while the bimodal encoding has almost the same accuracy (0.04% higher). On the other two datasets, the transformations improved the categorization accuracy by up to 1.76% and 1.00%, respectively. Given the fact that all three transformations incur only negligible computational costs and are very easy to implement, we recommend that they should be adopted in the encoding of action videos, especially the bimodal encoding.

Comparing with bag-of-features, the VLAD encoding is obviously more effective.[4] The improvements by the proposed transformations are relatively smaller when comparing with the BOF-VLAD shift, but are not trivial. For example, in the Youtbue dataset, a paired t-test shows that the improvement of even-scale is significant with p-value 0.0268. The bimodal encoding rates also compare favorably with the state-of-the-art results on these datasets, as shown in the third block.

4.3. Compress high-dimensional vectors

VLAD and FV vectors are usually very high dimensional (*e.g.*, when $K = 256$, VLAD is around 51200 dimensions in this paper). When a large number of videos are available (*e.g.*, UCF101 has 13320 videos), memory storage and CPU time requirements might make VLAD and FV become too costly for practical use. Many feature compression methods have been proposed to handle this problem (*e.g.*, product quantization [18]).

However, since there are far more images than videos, we probably do not need a very large compression ratio or sophisticated feature compression methods in the video domain. Two encodings on top of bimodal can compress VLAD with satisfactory results:

$$\text{1-bit} : x \leftarrow \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}, \tag{5}$$

$$\text{2-bit} : x \leftarrow \begin{cases} 1 & x > 0.4 \\ 0.5 & 0 \leq x \leq 0.4 \\ -0.5 & -0.4 \leq x < 0 \\ -1 & x < -0.4 \end{cases}. \tag{6}$$

They quantize a real-value into 2 and 4 discrete values (*i.e.*, 1 and 2 bits to store), respectively, corresponding to 32 and 16 times compression. The 1-bit encoding is the sign code in [15], and 2-bit is a simple extension of 1-bit. As shown in the second block of Table 4, although 1-bit rates are not satisfactory, 2-bit (with bimodal encoding) is close to VLAD without the encodings proposed in this paper.

Note that $x = 0$ is quantized into discrete value 1 or 0.5 in 1-bit and 2-bit, respectively. If we use 5 discrete values (2-bit plus $x = 0$, which requires 3 bits though), the accuracy is increased to 82.94% for UCF101 and 55.49% for HMDB51.

5. Conclusions

We empirically evaluated various improvement techniques for VLAD based video encoding, proposed a novel maxent interpretation, and proposed new encoding techniques to improve video encoding for action recognition. As a summary of our evaluations and findings, we suggest the following good practices for action video encoding:

- Use VLAD or FV instead of bag-of-features;
- Do not use large codebook size or large number of regions if your data is not complex. $K = 256$ and a small STP seems appropriate.
- Always use power normalization (because it changes unimodal features into bimodal, hence increasing their discriminative power), but only use root descriptors if it can improve normality;
- Use the bimodal encoding proposed in this paper. It is

[4] First row in the third block. HMDB51 and Youtube results are from [24]. For UCF101, we used $K = 4000$ in bag-of-features and the PmSVM classifier [26] with the histogram intersection kernel ($p = -16$, $C = 0.1$ in PmSVM).

simple, light-weight, yet effective;

- Use the 2-bit compression proposed in this paper if the dataset has a large number of videos.

Some future research topics are already discussed. For example, instead of using PCA for dimension reduction, we want to find maximum entropy linear feature function, which may improve the feature quality and consequently recognition accuracy. The bimodal encoding does not use the side information ($x = 0$), which can be further improved. We also want to extend our empirical evaluations to the Fisher Vector and to more benchmark datasets.

There is one final note and future topic: applying what we learned from encoding videos back to encode images. For example, we would like to test how the bimodal encoding will affect image encoding results.

Appendix

Given $\boldsymbol{x} \sim N(\mathbf{0}, \Sigma)$ (*i.e.*, a multivariate zero-mean normal distribution), we want to find a *maxent* linear projection $\boldsymbol{w}$ such that $\|w\| = 1$ and $y = \boldsymbol{x}^T \boldsymbol{w}$ has the maximum differential entropy $H(y) = -\int p(y) \log p(y)\, dy$.

Using properties of the normal distribution, it is easy to derive that $y \sim N(0, \boldsymbol{w}^T \Sigma \boldsymbol{w})$. Since the entropy of a standard normal distribution $N(0, \sigma^2)$ is $\frac{1}{2} \ln \left(2\pi e \sigma^2\right)$, we have that $H(y) = \frac{1}{2} \ln \left(2\pi e \boldsymbol{w}^T \Sigma \boldsymbol{w}\right)$. Thus, finding maxent linear projections for zero-mean Gaussian data is equivalent to solve the following PCA problem:

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T \Sigma \boldsymbol{w} \quad \text{s.t.} \quad \|\boldsymbol{w}\| = 1. \tag{7}$$

The generalization to multiple linear projections is trivial if we require the linear projections are perpendicular to each other—which is exactly the PCA requirement.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, pages 2911–2918, 2012.

[2] J. M. Chambers, W. S. Cleveland, P. A. Tukey, and B. Kleiner. *Graphical Methods for Data Analysis*. Duxbury Press, 1983.

[3] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez. Revisiting the VLAD image representation. In *ACM Multimedia*, 2013.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, Aug 2008.

[5] R. He, B. Hu, X. Yuan, and W.-S. Zheng. Principal component analysis based on non-parametric maximum entropy. *Neurocomputing*, 73(10-12):1840–1852, 2010.

[6] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, pages 2555–2562, 2013.

[7] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local images descriptors into compact codes. *IEEE TPAMI*, 34(9):1704–1716, 2012.

[8] Y.-G. Jiang, J. Liu, A. Roshan Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/ICCV13-Action-Workshop/`, 2013.

[9] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.

[10] I. Laptev. On space-time interest points. *IJCV*, 64(2/3):107–123, 2005.

[11] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[12] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, pages 1996–2003, 2009.

[13] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[14] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, pages 1817–1824, 2013.

[15] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, pages 3384–3391, 2010.

[16] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, LNCS 6314, pages 143–156, 2010.

[17] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, pages 1234–1241, 2012.

[18] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[19] M. Sapienza, F. Cuzzolin, and P. H. Torr. Learning discriminative spacetime action parts from weakly labelled videos. *IJCV*, to appear, 2014.

[20] K. Schindler and L. van Gool. Action snippets: How many frames does human action recognition require? In *CVPR*, 2008.

[21] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *ACM Multimedia*, pages 357–360, 2007.

[22] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human action classes from videos in the wild. Technical report, CRCV-TR-12-01, University of Central Florida, 2012.

[23] C. Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. In *IEEE Workshop on the Applications of Computer Vision*, pages 15–22, 2013.

[24] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, May 2013.

[25] X. Wang, L. Wang, and Y. Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In *Proc. Asia Conf. Computer Vision*, 2012.

[26] J. Wu. Power Mean SVM for large scale visual classication. In *CVPR*, pages 2344–2351, 2012.