Efficient Action Localization with Approximately Normalized Fisher Vectors

Dan Oneata, Jakob Verbeek, Cordelia Schmid

Efficient Action Localization with Approximately Normalized Fisher Vectors

Dan Oneata Jakob Verbeek Cordelia Schmid
Inria*

Abstract

The Fisher vector (FV) representation is a high-dimensional extension of the popular bag-of-word representation. Transformation of the FV by power and ℓ_2 normalizations has shown to significantly improve its performance, and led to state-of-the-art results for a range of image and video classification and retrieval tasks. These normalizations, however, render the representation non-additive over local descriptors. Combined with its high dimensionality, this makes the FV computationally expensive for the purpose of localization tasks. In this paper we present approximations to both these normalizations, which yield significant improvements in the memory and computational costs of the FV when used for localization. Second, we show how these approximations can be used to define upper-bounds on the score function that can be efficiently evaluated, which enables the use of branch-and-bound search as an alternative to exhaustive sliding window search. We present experimental evaluation results on classification and temporal localization of actions in videos. These show that the our approximations lead to a speedup of at least one order of magnitude, while maintaining state-of-the-art action recognition and localization performance.

1. Introduction

Recognition of human actions and activities is one of the most important topics in automatic video analysis. Two of the most generic and canonical problems in this area are the classification and localization of human actions. In classification the goal is to determine for a short video clip whether or not it contains a given human action of interest [31]. For localization the goal is to report in a longer video all (spatio-) temporal windows that contain the action of interest [17]. These problems are intimately related, and are counterparts in the video domain of the widely studied image classification and object detection problems in still images [3, 6]. With the advent of advanced local spatio-temporal features and feature encoding techniques, research on action recognition and localization has recently made significant progress, and moved on from simple controlled video datasets [26], to address these tasks on uncontrolled videos extracted from movies or YouTube [15, 20].

The Fisher vector (FV) image representation is an extension of the popular bag-of-visual-word (BoV) representation, that yields high-dimensional image signatures by encoding for each visual word the mean and variance of the assigned local descriptors [22]. With the inclusion of power and ℓ_2 normalizations proposed by Perronnin *et al*. [23], the FV image representation has proven to be one of the most effective ones for image classification, see *e.g*. the evaluation study of Chatfield *et al*. [3]. Recently the FV has appeared for a wide variety of problems as the representation that underpins state-of-the-art results, including image retrieval [14], object detection [6], and semantic segmentation [19]. Also for action recognition and localization FVs have recently been explored, and shown to yield state-of-the-art performance, see *e.g*. [21, 27, 33]. In particular the combination of motion-based descriptors computed along densely sampled temporal feature tracks [31] with FV encoding is currently one of the most effective representations.

To a large extent, the success of the FV representation can be ascribed to its high dimensionality, which makes it very effective in combination with efficient linear classifiers. The flip side of its high dimensionality — representations of tens to hundreds of thousands of dimensions are more the rule than an exception— is, however, that it leads to large storage requirements. This issue becomes particularly pressing in large-scale image classification and retrieval tasks. Data compression with product quantization has been employed to reduce the storage requirements by orders of magnitude in this context, see *e.g*. [14, 24].

Localization of actions in video, or objects in images, can be considered as a large-scale classification problem, where we want to find the highest scoring windows in a video or image w.r.t. a classification model of the category of interest. Unlike generic large-scale image classification, however, the problem is highly structured in this case, in the sense that all windows are crops of the same video or image under consideration. This structure has been exten-

*LEAR team, Inria Grenoble Rhône-Alpes, Laboratoire Jean Kuntzmann, CNRS, Univ. Grenoble Alpes, France

sively exploited in the past. In particular, when the features for a detection window are obtained as sums of local features, integral images can be used to pre-compute cumulative feature sums. Once the integral images are computed, these can be used to compute the sums of local features in constant time w.r.t. the window size. Viola and Jones [30] used this idea to efficiently compute Haar filters for face detection. Recently, Chen *et al*. [4] used the same idea to aggregate scores of local features in an object detection system based on a non-normalized FV representation. Another way to exploit the structure of the localization problem is to use branch-and-bound search, as *e.g*. used by Lampert *et al*. [16]. Instead of evaluating the score of one window at a time, they hierarchically decompose the set of detection windows and consider upper-bounds on the score of sets of windows to explore the most promising ones first.

While the power and ℓ_2 normalizations of Perronnin *et al*. [23] have proven effective to improve the performance of the FV, the resulting normalized FV is no longer additive over local features. As a consequence, these FV normalizations prevent the use of integral image techniques to efficiently aggregate local features or scores.

Our first contribution, which we present in Section 3, is to show that the FV normalizations can be approximated in a way that the score for an arbitrarily large window can be computed in constant time, by relying on pre-computed cumulative sums of local visual word assignments, scores, and ℓ_2 norms. Second, in Section 4, we show that with our approximations the normalized FV becomes amenable to efficient localization using branch-and-bound search.

In our experimental evaluation, presented in Section 5, we validate on two action classification benchmarks that our approximations have only a limited impact on the effectiveness of the normalizations. Experiments on two temporal action localization datasets demonstrate that our approximations accelerate temporal localization by more than one order of magnitude when using a temporal sliding window approach. Branch-and-bound search brings further speedup when only the top scoring windows need to be reported. Before presenting our contributions and experiments in detail, we first discuss some of the most relevant related work.

2. Related work

The search space for multi-dimensional localization problems is large: if we consider a D-dimensional grid with B bins per dimension, there are $C = B^D$ grid cells, and $O(C^2)$ windows defined over the grid. Exhaustive sliding window search is therefore costly, unless low-dimensional features in combination with linear classifiers are used.

Viola and Jones [30] introduced a face detector that combined efficient computation of Haar-filters over pixel intensities using integral images, with a detection-cascade that progressively rejects detection windows using an increasingly larger set of features. In this manner most computation is spent on finely analyzing the most promising image regions. Similar ideas were used by others for generic object category detection using richer image representations based on BoV representations, by using progressively more expensive classifiers, see *e.g*. [11, 29].

If an additive window representation is used —such as a non-normalized BoV histogram— in combination with a linear classifier, several efficient algorithms are available for localization. These algorithms exploit the commutative property of the linear score function and the additivity of the representation. For the one-dimensional case the problem then reduces to the maximum subarray problem, which can be solved with a linear-time dynamic programming algorithm. In the two-dimensional case, An *et al*. [2] presented an $O(C^{3/2})$ algorithm, which was used by Chen *et al*. [4] for detection with non-normalized FVs. Another approach, used by Lampert *et al*. [16], is branch-and-bound search, for which the bounds are efficiently evaluated for additive features and linear classifiers. Yuan *et al*. [35] generalized this approach to spatio-temporal localization in videos.

Most of the recent work that uses FV representations for object and action localization, and semantic segmentation, either uses non-normalized FVs [4, 7], or explicitly computes normalized FVs for all considered windows [6, 21]. The recent work of Li *et al*. [19] is an exception to this trend; they left out the power-normalization of the FV, but presented an efficient approach to incorporate exact ℓ_2 normalization. In this paper we present approximations to both the power and ℓ_2 normalization, which allows us to compute the score of a window by aggregating locally pre-computed quantities. In particular, we store for each cell and visual word the local sum of assignments, scores, and ℓ_2 norms. This representation, therefore, has a size that is only three times larger than a local BoV histogram, while leveraging the representational power of the normalized FV.

A different line of work focuses on using category-independent selective search techniques, mainly driven by low-level contour and segmentation cues, to produce a small set of candidate detection windows, see *e.g*. [1, 28]. In this manner a set of only 1,000 to 2,000 windows suffices to capture 95% of the objects in the PASCAL VOC datasets. Since these techniques are decoupled from the actual detector, they do not impose any constraints on the detector or its features. Such techniques can be generalized to the video domain by using efficient super-voxel techniques, see *e.g*. [8, 34]. Our approximate normalizations can be used in combination with such selective search techniques.

3. Approximate Fisher vector normalization

Below, we first briefly review the Fisher vector image representation, after which we present our approximations to the power and ℓ_2 normalization. Finally, we analyze the

complexity to compute the approximately normalized FV.

3.1. The Fisher vector and its normalizations

The Fisher kernel principle of Jaakkola and Haussler [12] uses generative models for feature extraction by representing data by means of the gradient of the data log-likelihood w.r.t. the model parameters. They showed that this representation becomes invariant to re-parametrization of the generative model when the gradients are normalized by the inverse-square-root of the Fisher information matrix.

Perronnin *et al*. [22] applied the Fisher kernel principle to obtain image representations based on sets of N local features, *e.g*. SIFTs, which they modeled as independent samples from a K-component Gaussian mixture model (GMM). They used Gaussians with diagonal covariance matrices, and we use the vector σ_k to denote these diagonals. Let $x_n \in \mathbb{R}^d$ denote the n-th d-dimensional local feature, q_{nk} the soft-assignment of x_n to the k-th Gaussian, and π_k and μ_k the mixing weight and mean of the k-th Gaussian respectively. The d-dimensional gradients w.r.t. the mean and variance of the k-th Gaussian are given by:

$$G_{\mu_k} = \sum_{n=1}^{N} q_{nk} \left[x_n - \mu_k\right] / \sqrt{\sigma_k \pi_k}, \quad (1)$$

$$G_{\sigma_k} = \sum_{n=1}^{N} q_{nk} \left[(x_n - \mu_k)^2 - \sigma_k\right] / \sqrt{2\sigma_k^2 \pi_k}, \quad (2)$$

where operations using σ_k should be understood as element-wise operations, and the normalization by the Fisher information matrix has already been taken into account. The concatenation of these d dimensional gradients, as $G = [G_1, \ldots, G_K]$ with $G_k = [G_{\mu_k}, G_{\sigma_k}]$, is then referred to as the Fisher vector (FV), which is of dimension $2Kd$. The gradient w.r.t. the mixing weights of the GMM are generally ignored since they contribute little discriminative power to the FV. See [25] for a recent comprehensive review of the FV image representation.

Two normalizations of the FV representation significantly improve its performance [23]. The first of these is the power normalization which consists in applying per-dimension a "signed" power, by transforming each element of the FV as $z \leftarrow \text{sgn}(z)\text{abs}(z)^\rho$, with $0 < \rho < 1$. The second normalization is the ℓ_2 normalization, which consists in rescaling the FV to have unit ℓ_2 norm.

3.2. Approximate power normalization

Cinbis *et al*. [5] have argued that the power normalization corrects for the independence assumption that is made in the GMM model that underpins the FV representation. They presented latent variable models which do not make this independence assumption, and experimentally found that such models lead to similar performance improvements as the power-normalization. In particular, they showed that the gradients w.r.t. the mixing weights in their non-i.i.d. model take the form of discounted version of these gradients in the original i.i.d. model. The transformation they found was the di-gamma function, which, like the power-normalization, is a concave monotonic function.

Based on this analysis, we propose an approximate version of the power normalization. First, note that the gradients in G_k are *weighted sums* of contributions of local features. Let us write these in a more compact manner as:

$$G_k = \sum_n q_{nk} g_{nk} = \left(\sum_n q_{nk}\right) \sum_n \frac{q_{nk} g_{nk}}{\sum_m q_{mk}}, \quad (3)$$

where q_{nk} and g_{nk} denote the weight and gradient contribution of the n-th local descriptor for G_k. The last part of Eq. (3) re-interprets the FV as a *weighted average* of local contributions, multiplied by the sum of the weights. The power-normalization is computed as an element-wise signed-power of G_k. In our approximation we, instead, apply the power only to the (positive) sum of weights:

$$\mathcal{G}_k = \left(\sum_n q_{nk}\right)^\rho \sum_n \frac{q_{nk} g_{nk}}{\sum_m q_{mk}}. \quad (4)$$

In our approximation, the power-normalization modifies the magnitude of the gradient vector, but not its orientation. We concatenate the $\mathcal{G}_k$ to form the normalized FV $\mathcal{G}$.

Using our approximate power-normalization, a linear function can now be computed by aggregating local scores. For a classifier weight vector $w = [w_1, \ldots, w_k]$ we have:

$$\langle w, \mathcal{G} \rangle = \sum_k \langle w_k, \mathcal{G}_k \rangle = \sum_k \left(\sum_n q_{nk}\right)^{\rho-1} \sum_n s_{nk}, \quad (5)$$

where $s_{nk} = \langle w_k, q_{nk} g_{nk} \rangle$ denotes the score of the local non-normalized FV, which is still additive over local terms.

3.3. Approximate ℓ_2 normalization

We now proceed with an approximation of the ℓ_2 norm of $\mathcal{G}$. The squared ℓ_2 norm is a sum of squared ℓ_2 norms per Gaussian component: $||\mathcal{G}||_2^2 = \sum_k \mathcal{G}_k^\top \mathcal{G}_k$, where

$$\mathcal{G}_k^\top \mathcal{G}_k = \left(\sum_n q_{nk}\right)^{2(\rho-1)} \sum_{n,m} q_{nk} q_{mk} \langle g_{nk}, g_{mk} \rangle . \quad (6)$$

We approximate the double sum over dot products of local gradient contributions by assuming that most of the local gradients will be near orthogonal for high-dimensional FVs. This leads to an approximation $L(\mathcal{G}_k)$ of the squared ℓ_2 norm of $\mathcal{G}_k$ in the form of a sum of local contributions:

$$L(\mathcal{G}_k) = \left(\sum_n q_{nk}\right)^{2(\rho-1)} \sum_n q_{nk}^2 l_{nk}, \quad (7)$$

where $l_{nk} = \langle g_{nk}, g_{nk} \rangle$ is the local squared ℓ_2 norm. Summing these over the visual words, we approximate $||\mathcal{G}||_2^2$ with $L(\mathcal{G}) = \sum_k L(\mathcal{G}_k)$.

Theoretically, we can show the following: (i) In expectation the approximation of ℓ_2 norm converges to the true ℓ_2 norm if the contributing descriptors are independently distributed. (ii) If the FV of local descriptors are positively correlated (which we expect them to be in practice), our approximated L2 norm is an underestimate of the true norm. We will asses the validity of these properties in practice in our experiments in Section 5.

3.4. Complexity of approximately normalized FVs

We combine the above approximations to compute a linear function of our approximately normalized FV as

$$f(\mathcal{G}; w) = \left\langle w, \mathcal{G}/\sqrt{L(\mathcal{G})} \right\rangle = \langle w, \mathcal{G} \rangle \Big/ \sqrt{L(\mathcal{G})}. \quad (8)$$

To efficiently compute $f(\mathcal{G}; w)$ over many windows of various sizes and positions, we can use integral images to compute in constant time the per-window sums of assignments, scores, and norms in s, q, and l. Since the assignments, scores, and norms vary per visual word, we need to compute three integral images for each visual word.

For the complexity analysis we again assume we use a multidimensional grid, with C cells in total. When either using the exact or our approximate FV normalizations, the first step is to aggregate the local FVs per cell, and to compute the (multi-dimensional) integral image over these cells. When using our approximations, we will compute $3K$ integral images that accumulate the local weights, scores, and norms per visual word. For exact normalization we need to compute $2Kd$ integral images, since we first need to compute the full window-level FVs before the normalizations can be applied. The cost of this step is in both cases $O(CKd)$. As compared to storing the full local FV, our representation is $2d/3$ times more compact. When using, as in our experiments, $d = 192$ for the local features, this reduces the storage requirements by a factor $2 \times 192/3 = 128$.

Once the integral images are available, the cost to score a window of arbitrary size is $O(K)$ with our approximations, as opposed to $O(Kd)$ when using exact normalization. We thus obtain an $O(d)$ speedup for the window scoring.

The actual cost to obtain the integral images is slightly higher when using our approximate normalizations, since at this stage we already compute local scores and norms, and take powers of the weight sums. When using exact normalization, we only sum local FVs at this stage. This cost is, however, amortized as using our approximations the per-window scoring is a factor d faster; and exact normalization requires taking powers of the full window-level FVs. In our experiments we assess the speedup as observed in different practical settings, as well as the impact of the normalizations on the recognition performance.

4. Integration with branch-and-bound search

The approximations we presented above accelerate the scoring of windows by aggregating locally pre-computed scores, weights, and norms. A second method to speedup detection is to use a branch-and-bound search instead of exhaustive search. The idea of branch-and-bound is to evaluate upper bounds on the scores of windows in sets of detection windows. Starting from the set of all possible windows, the search is organized by hierarchically splitting sets of windows and computing upper bounds on the scores. A set of detection windows is represented by a tuple $\mathcal{A} = (s_{low}, s_{high}, e_{low}, e_{high})$, where $s_{low} \in \mathbb{R}^D$ defines the earliest starting point for the windows in $\mathcal{A}$ on the D-dimensional grid. Similarly, s_{high} defines the latest starting point for all windows in $\mathcal{A}$, and e_{low} and e_{high} define the earliest and latest end points. Sets of windows are split by either splitting the start range or the end range on a single dimension, depending where the range is maximum. The sets are explored in a best-first manner, which focuses on the most promising parts of the search space first. See [16] for a comprehensive introduction to branch-and-bound search.

Below we present an upper bound for the score defined in Eq. (8). For clarify of exposition, we start with a bound on a simple additive score function, and then present bounds when adding our approximate power and ℓ_2 normalizations.

4.1. Upper-bound for additive linear classifiers

If the function is linear and additive in the local features it is easy to obtain an upper bound by separating the positive and negative terms. In particular, consider such a function over the non-normalized FV G:

$$f(G; w) = \langle w, G \rangle = \sum_k \langle w_k, G_k \rangle = \sum_k \sum_n s_{nk},$$
$$= \sum_k \left[\sum_{n: s_{nk} > 0} s_{nk} + \sum_{n: s_{nk} < 0} s_{nk} \right], \quad (9)$$

where as before $s_{nk} = \langle w_k, q_{nk} g_{nk} \rangle$. We can upper bound the score of the FV of any window in a set of windows $\mathcal{A}$ by accumulating positive and negative scores of the union $\mathcal{A}_\cup$ and intersection $\mathcal{A}_\cap$ of all windows in $\mathcal{A}$ respectively:

$$f(\mathcal{A}; w) = \sum_k \left[\sum_{n \in \mathcal{A}_\cup : s_{nk} > 0} s_{nk} + \sum_{n \in \mathcal{A}_\cap : s_{nk} < 0} s_{nk} \right]. \quad (10)$$

4.2. Bounding approximate power-normalization

When using our approximate power-normalization, a linear classification score takes the form of Eq. (5):

$$f(\mathcal{G}; w) = \langle w, G \rangle = \sum_k \left(\sum_n q_{nk} \right)^{\rho - 1} \sum_n s_{nk}. \quad (11)$$

To bound this function for a set of windows $\mathcal{A}$, we can use the previous bound for the linear score terms $\sum_n s_{nk}$. Provided the intersection $\mathcal{A}_\cap$ is non-empty, the scalar multiplication with $(\sum_n q_{nk})^{\rho-1}$ can be bounded by accumulating the weights over the intersection $\mathcal{A}_\cap$ instead. For $0 < \rho < 1$ this leads to the upper bound

$$f_1(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cap} q_{nk} \right)^{\rho-1} \left[\sum_{\substack{n \in \mathcal{A}_\cup \\ s_{nk} > 0}} s_{nk} + \sum_{\substack{n \in \mathcal{A}_\cap \\ s_{nk} < 0}} s_{nk} \right].$$

If the intersection $\mathcal{A}_\cap$ is empty, however, we obtain a trivial upper bound $f(\mathcal{A}) = \infty$, since $0^{(\rho-1)} = \infty$ for $0 < \rho < 1$. To bound this case, we use the interpretation of the normalized FV as a weighted average, *c.f.* Eq. (4), and write:

$$f(\mathcal{G}; w) = \sum_k \left(\sum_n q_{nk} \right)^{\rho} \sum_n \frac{q_{nk} \langle w_k, g_{nk} \rangle}{\sum_m q_{mk}}. \quad (12)$$

It is then easy to see that we can upper bound the score as

$$f_2(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^{\rho} \max_{n \in \mathcal{A}_\cup} \langle w_k, g_{nk} \rangle. \quad (13)$$

Since the latter bound relies on a non-linear max operation, we cannot efficiently compute it using intergral images. Therefore, the bound $f_1(\mathcal{A})$ is preferred if $\mathcal{A} \neq \emptyset$.

4.3. Bounding with approximate ℓ_2 norm included

To upper bound a linear function of our approximately power and ℓ_2 normalized FVs, *c.f.* Eq. (8), for a set of windows $\mathcal{A}$, we need to lower bound the approximate ℓ_2 norm:

$$L(\mathcal{G}) = \sum_k \left(\sum_n q_{nk} \right)^{2(\rho-1)} \sum_n q_{nk}^2 l_{nk}. \quad (14)$$

Since $2(\rho - 1) < 0$, the first term can be bounded summing over the union $\mathcal{A}_\cup$ instead. If the intersection $\mathcal{A}_\cap$ is non-empty we can bound the second term by summing over the intersection, and obtain the lower-bound on $L(\mathcal{G})$ as:

$$L_1(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^{2(\rho-1)} \sum_{n \in \mathcal{A}_\cap} q_{nk}^2 l_{nk}. \quad (15)$$

If the intersection is empty, we can instead of the sum over $\mathcal{A}_\cap$, use the minimum over $\mathcal{A}_\cup$ to obtain the bound:

$$L_1(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^{2(\rho-1)} \min_{n \in \mathcal{A}_\cup} q_{nk}^2 l_{nk}. \quad (16)$$

It is easy to verify that if $\mathcal{A}_\cap \neq \emptyset$ then $L_2(\mathcal{A}) \leq L_1(\mathcal{A})$, thus in this case $L_1(\mathcal{A})$ is the tightest of the two bounds.

5. Experiments

Below, we first discuss the datasets, features, and evaluation protocols in Section 5.1. Then, in Section 5.2, we present results on action classification to evaluate the impact of our approximate FV normalizations on recognition performance. We evaluate the speedup our approximations brings to temporal action localization in Section 5.3.

5.1. Datasets, evaluation protocols, and features

Below we describe the datasets used in our experiments. **Hollywood2** [20] contains samples collected from Hollywood movies of 12 action categories. There are 810 and 884 train and test clips respectively, and these sets are selected from different movies. The average duration of clips is about 20 sec. We use the standard evaluation protocol and report the mean average precision (mAP) across the actions.

HMDB [15] contains video clips of 51 action categories. We follow the standard evaluation protocol, and report the average classification accuracy over three test-train splits that per category contain 70 examples for training, and 30 for testing. We use the non-stabilized version of the videos.

Coffee and Cigarettes is an action localization dataset [17] annotated with instances of two actions: *drinking* and *smoking*. The movie consists of 11 short stories, each with a different set of actors. Train and test sets are taken from different short stories; additional training examples are included from the movie *Sea of Love* as well as lab-recorded ones. For drinking there are 106 training samples, while the test set consists of 20 minutes of video that contains 38 positive samples. For *smoking* there are 78 training samples, and the test set consists of 18 minutes of video that contains 42 positive samples.

The **Duchenne** dataset [9] contains annotations for the actions *sit down* and *open door*. The training data comes from 15 movies, and contains 51 *sit down* examples, and 38 for *open door*. The test data contains three full movies (*Living in Oblivion*, *The Crying Game*, and *The Graduate*), which in total last for about 250 minutes, and contain 86 *sit down*, and 91 *open door* samples.

To evaluate localization performance we follow the standard protocol [9, 17], and report the average precision (AP), where we consider a localization correct if the temporal window overlaps with a ground-truth action by at least 20% in the sense of intersection-over-union. For localization we consider temporal windows with lengths from 20 to 180 frames, with increments of 5 frames. We use a stride of five frames to locate the windows on the video. We use zero-overlap non-maximum suppression, and re-scale the window scores by the duration, as in [21]. When using branch-and-bound, window sets that are guaranteed to intersect already selected windows are removed from the queue.

Features. We use the public implementation of the dense trajectory features of Wang *et al*. [31], with standard

Power norm.	ℓ_2 normalization	Hollywood2	HMDB
No	No	55.2	43.1
Exact	No	62.0	51.7
No	Exact	60.1	46.8
Exact	Exact	62.4	52.2
Approximate	Exact	62.1	52.1
Approximate	Approximate, $n = 5$	60.1	52.6
Approximate	Approximate, $n = 10$	60.2	52.4
Approximate	Approximate, $n = 20$	60.2	52.6
Approximate	Approximate, $n = 40$	60.6	52.5
Approximate	Approximate, $n = 80$	60.7	52.2
Approximate	Approximate, $n = 160$	61.1	52.2
Wang *et al*. 2013	[31]	59.9	48.3
Oneata *et al*. 2013	[21]	61.9	51.9
Jain *et al*. 2013	[13]	62.5	52.1
Wang *et al*. 2013	[32]	64.3	57.2

Table 1. Action classification performance. For the ℓ_2 approximation we evaluate using cells of n frames, for $n = 5$ to $n = 160$.

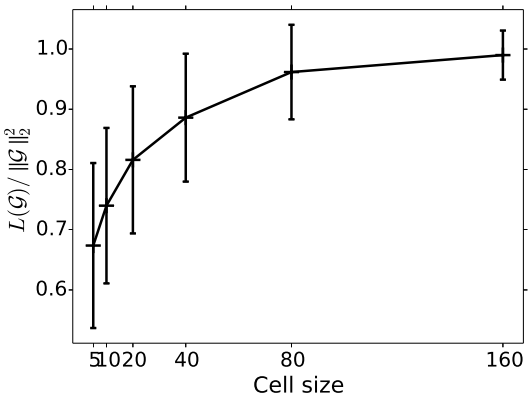

Figure 1. Errors in the ℓ_2 norm approximation, see text for details.

parameters. We extract MBH features and project them to 64 dimensions with PCA. As in Oneata *et al*. [21], we use 1,000 GMM components for classification, and include position information with spatial pyramids and spatial Fisher vectors. For temporal localization we use a GMM with 128 components, and no position information. This setting yields a FV of 804,000 dimensions for classifications and 16,384 for localization.

Classifiers. We use linear SVM classifiers, and for the multi-class datasets Hollywood2 and HMDB we use a one-versus-rest approach. We cross-validated the regularization parameter and the class balancing weight.

5.2. Effect of approximation on action classification

In our first experiment we consider the effect of the power and ℓ_2 normalizations of the FV for action classification, and assess to which degree our approximations maintain the performance benefits of the exact normalizations. In our experiments we use the common setting of $\rho = \frac{1}{2}$, see *e.g*. [3, 14, 21], which corresponds to a signed square-root.

The first four results in Table 1 assess the effectiveness of the exact power and ℓ_2 normalization for action classification. For both datasets the power normalization is the most effective one, improving performance by 6.8 and 8.6 mAP points respectively. Adding ℓ_2 normalization improves results further by 0.4 and 0.5 mAP points respectively.

When using approximate power normalization (fifth line), but exact ℓ_2 normalization, performance drops only slightly for both datasets. For Hollywood2 and HMDB the loss is only 0.3 and 0.1 points respectively; which is respectively 2.0 and 5.3 points above not using power normalization. Experimentally, we found that it is beneficial to apply an additional element-wise standardization of the FV after approximate power normalization, and before ℓ_2 normalization. If this is not done, performance drops by about 1 point to 61.3% and 51.1% mAP respectively. This standardization can absorbed in the classifier weight vector w, before computing the local scores s_{nk}, and therefore does not impact the computational efficiency of our approach.

The following six results show the effect of approximate ℓ_2 normalization for various temporal cell sizes over which the features are aggregated: from 5 up to 160 frames. The smaller the cell size, the coarser our approximation, since more cross-terms will be ignored in our approximation. The results show, however, that the classification performance is only slightly impacted by using smaller cells. For Hollywood2 the best result of 61.1% is obtained for cells of 160 frames, while using 5-frame cells yields 60.1% mAP. For HMDB the variation in performance across different cell sizes is at most 0.4 points, with better performance for smaller cells.

We further analyze the ℓ_2 norm approximation by considering the ratio between the approximate and the true norm. In Figure 1 we show the average of the ratio and its standard deviation for various cell sizes measured on HMDB. Note that the ratio is generally smaller than one, as expected. Moreover, even for small cell sizes the under estimation is limited to about a factor two, with limited variation in the under estimation factor. This explains the small impact on the recognition performance observed above.

To show that our video representation is competitive with the state-of-the-art, we include four recent state-of-the-art results of [13, 21, 31, 32]. Wang *et al*. [31] use a sum of RBF chi-squared kernels over BoV histograms for HOG, HOF, and MBH features, computed over six different SPM grids [18], and using 4,000 visual words. The setup of Oneata *et al*. [21] is comparable to the one we used here. Jain *et al*. [13] use camera motion stabilization before computing MBH, HOG, and HOF features, in addition to their DCS flow features. They aggregate these features using VLAD descriptors [14], a variant of FV. Wang *et al*. [32]

Search	Power	ℓ_2	Drinking	Smoking	Open Door	Sit Down
Exh.	No	No	34.0	15.6	10.3	5.9
Exh.	No	Yes	61.1	55.5	24.1	18.3
Exh.	Yes	No	64.8	23.8	20.6	17.1
Exh.	Yes	Yes	64.8	55.4	28.4	19.0
Exh.	Appr.	Appr.	67.1	52.0	18.1	13.6
Oneata *et al*.		[21]	63.9	50.5	26.5	18.2
Gaidon *et al*.		[10]	57	31	16.4	19.8
Laptev & Pérez		[17]	49	—	—	—
Duchenne *et al*.		[9]	40	—	14.4	13.9

Table 2. Action localization performance using either no, exact, or approximate normalizations, and recent state-of-the-art results.

also use stabilized features, but they estimate a homography using RANSAC and then improve their matches using a human detector; these features are then encoded with FV.

5.3. Temporal action localization

In our temporal action localization experiments, we compare using exact normalizations and our approximations in terms of localization performance and speed.

The localization results are presented in Table 2. As before, the first four results consider the impact of the exact normalizations on performance. For *drinking* and *sit down* the power and ℓ_2 normalization have a similar impact, and improve the results by about 30 and 12 mAP points respectively. Using both normalizations does not bring further improvements for *drinking*, but does improve results by 0.7 points for *sit down*. For *smoking* and *open door* the ℓ_2 normalization brings the largest improvement of almost 40 and 14 mAP points respectively. Additional power normalization is not effective for *smoking*, but does bring an improvement of 4.3 points for *open door*.

Next, we consider the impact of our approximate power and ℓ_2 normalizations. For *drinking* we obtain an AP of 67.1%, which is even 2.3 points above the results for exact normalization. For *smoking* our approximations also lead to an AP of 52.0%, which is 3.4 points below the result for exact normalization. For *open door* and *sit down* the results are 10.3 and 5.4 points below those obtained using exact normalization. They are, however, still 7.8 and 7.7 points better than not using normalization. For *drinking* and *smoking* the gain of our approximate normalization w.r.t. no normalization is 33.1 and 36.4 mAP points.

We include recent state-of-the-art results of [9, 10, 17, 21] to show that our results are competitive. The results for Laptev *et al*. [17] are taken from [9], which have interpolated their spatio-temporal localization results to the temporal domain. Our results with exact normalization are above the best earlier reported results. Using our approximations

Search	Power	ℓ_2	Drinking	Smoking	Open Door	Sit Down
Exh.	No	No	2.8	2.2	31.1	29.9
Exh.	No	Yes	95.4	92.2	1276.7	1168.3
Exh.	Yes	No	146.3	143.4	1794.3	1781.9
Exh.	Yes	Yes	160.3	151.0	1966.8	2036.4
Exh.	Appr.	Appr.	11.3	10.3	140.6	138.0

Table 3. Timings (secs.) for action localization using exhaustive search with either no, exact, or approximate normalizations.

	Drinking	Smoking	Open Door	Sit Down
Pre-processing	7.65	6.99	93.01	92.98
Exhaustive search	3.65	3.31	47.67	43.63
Branch and bound, top 1	1.02	1.40	16.59	20.57
Branch and bound, top 10	1.71	2.44	35.41	30.32

Table 4. Search times for the action detection datasets in seconds for exhaustive and branch-and-bound search, where both use our approximate normalization.

this is still the case for *drinking* and *smoking*, but not for *open door* and *sit down*.

The timing results in Table 3 show that our approximations lead to a speedup of about one order of magnitude w.r.t. using exact normalization. As compared to using no normalization, our approximations are about four to five times slower, but leads to significantly better results.

Finally, we evaluate the speed of branch-and-bound search with our approximations to find the top scoring temporal windows. In Table 4 we give an overview of the search times when searching for the t highest scoring windows, for t=1 and t=10. Both exhaustive and branch-and-bound search first pre-process all the cells in the temporal grid to compute the local sums of scores, assignments, and norms. We separate the time needed for pre-processing, and the actual search time. When searching for the top window, branch-and-bound is between 2.1 and 3.6 times faster. For the top 10 windows, the speedup factors are between 1.3 and 2.4. Although faster than exhaustive search, overall the speedup obtained using branch-and-bound is limited. This is because in our uni-dimensional temporal search setup, the number of windows is only 32 times larger than the number of cells in the search grid. We expect larger speedup for branch-and-bound when applied to 2D spatial, or 3D spatio-temporal localization problems.

6. Conclusion

We have presented approximate versions of the power and ℓ_2 normalization of the Fisher vector representation.

These approximations allow efficient evaluation of linear score functions, by caching local per visual word sums of scores, assignments, and norms. We also presented corresponding upper-bounds that permit the use of our approximations in branch-and-bound search.

Experimental results for action classification and localization show that these approximations only have a limited impact on performance, while yielding speedups of at least one order of magnitude. When only the top-scoring window is required, branch-and-bound search can further speedup localization by a factor between 2 to 4, excluding pre-processing.

The efficient localization techniques presented here are directly applicable to other localization tasks, such as object localization in still images, and spatio-temporal action localization. Since these tasks consider higher dimensional search spaces, we expect the speedup of our approximations, as well as branch-and-bound search, to be even larger than for temporal localization task that we considered in this paper. We plan to explore application of our approximations for these tasks in future work.

Acknowledgements. This work was supported by the European integrated project AXES, and the ERC advanced grant ALLEGRO.

References

[1] B. Alexe, T. Deselares, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 34(11):2189–2202, 2012.

[2] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *CVPR*, pages 264–271, 2009.

[3] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[4] Q. Chen, Z. Song, R. Feris, A. Datta, L. Cao, Z. Huang, and S. Yan. Efficient maximum appearance search for large-scale object detection. In *CVPR*, 2013.

[5] R. Cinbis, J. Verbeek, and C. Schmid. Image categorization using Fisher kernels of non-iid image models. In *CVPR*, 2012.

[6] R. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with Fisher vectors. In *ICCV*, 2013.

[7] G. Csurka and F. Perronnin. An efficient approach to semantic segmentation. *IJCV*, 95(2):198–212, 2011.

[8] M. V. den Bergh, G. Roig, X. Boix, S. Manen, and L. V. Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013.

[9] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.

[10] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[11] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[12] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[13] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

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

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

[16] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: a branch and bound framework for object localization. *PAMI*, 31(12):2129–2142, 2009.

[17] I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007.

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

[19] Z. Li, E. Gavves, K. van de Sande, C. Snoek, and A. Smeulders. Codemaps, segment classify and search objects locally. In *ICCV*, 2013.

[20] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009.

[21] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In *ICCV*, 2013.

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

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

[24] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

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

[26] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *ICPR*, 2004.

[27] C. Sun and R. Nevatia. Large-scale web video event classification by use of Fisher vectors. In *WACV*, 2013.

[28] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[29] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[30] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2004.

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

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

[33] X. Wang, L. Wang, and Y. Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In *ACCV*, 2012.

[34] C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[35] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009.