A Piggyback Representation for Action Recognition

Lior Wolf, Yair Hanani
The Balvatnik School of Computer Science
Tel Aviv University

Tal Hassner
Dept. of Mathematics and Computer Science
The Open University of Israel

Abstract

In video understanding, the spatial patterns formed by local space-time interest points hold discriminative information. We encode these spatial regularities using a word2vec neural network, a recently proposed tool in the field of text processing. Then, building upon recent accumulator based image representation solutions, input videos are represented in a hybrid manner: the appearance of local space time interest points is used to collect and associate the learned descriptors, which capture the spatial patterns. Promising results are shown on recent action recognition benchmarks, using well established methods as the underlying appearance descriptors.

1. Introduction

Understanding human actions in unconstrained 2D videos has been a central theme in Computer Vision research for decades, and considerable progress has been achieved. The gap between the current capabilities and the needs of real-world applications still remains wider than in other vision domains, including photo categorization, face recognition, occlusion avoidance, and pose-recognition in constrained depth video.

This gap in performance is not surprising, when one considers the many challenges that must be addressed in attempting to automatically decide what actions are being performed in a video. This is particularly true when one compares video analysis to image analysis problems. For one thing, images can often be assumed to have the cameraman controlling the appearance of the photo. Videos, on the other hand, contain many frames (many images) over which the cameraman has far less control. Individual frames can therefore be noisy, contain motions that are irrelevant to the action of interest, and the actors themselves may be partially or completely occluded at different parts of a video clip. All these issues are exacerbated by the sheer size of the videos and the processing and storage overhead that this can imply.

To make sense of these many challenges and confounding factors, action recognition research addresses four key aspects of action recognition, all of which are likely to be essential for successfully identifying actions in videos. These are:

(i) Appearance: The same action may be performed by different people in different settings, and so appear very different. Effective action recognition therefore requires developing representations that are invariant to these changes, yet capable of discriminating between different activities.

(ii) Spatial arrangement: Different feature arrangements can imply different actions. Representing arrangements of image features is therefore a crucial step towards representing the action itself.

(iii) Temporal arrangement: The order in time in which features appear captures much information about the action being performed. Moreover, the motion trajectories of local scene elements, features tracked in time, also hold substantial amounts of information.

(iv) Context: Where the action is taking place – the scene around the action – and how the camera is moving in that scene, are two examples of contextual information. This information can provide important clues for identifying the action being performed.

Our method focuses on the second aspect. However, as it attempts to capture spatial arrangements and their significance, it builds upon techniques that have been developed to capture the aspect of appearance. Although it is not evaluated in this work, the same approach can also be used to capture temporal arrangements. Our method builds upon several recent advancements to the Bag of Keywords methodology, in particular the Vector of Locally Aggregated Descriptors (VLAD) method of [7]. In order to capture local structures, it employs a variant of a method called word2vec [16] to encode the local spatial arrangements of keypoints in each frame. The resulting system seems to be unique in that it provides a global descriptor that captures the regularities of local structures, but does not try to detect those structures in the video. We demonstrate its capabilities by testing it on recent, challenging action recognition benchmarks.

Our main contributions are: (i) The application of log-linear neural nets to capture spatial relations is novel and

very different from existing solutions. (ii) The process of applying word2vec to images/video, as introduced here, including a spatial word2vec variant and VLAD based joining, is novel and not an obvious extension of any existing method. (iii) Our experiments show significant improvement in two challenging benchmarks. In one benchmark our improvement is greater than any previous performance leap on this benchmark.

2. Previous Work

Our work borrows from multiple domains including video analysis, visual keypoint representation, and natural language processing.

2.1. Video content representations.

Over the years many attempts have been made to design effective representations of video content. These range from high-level shape representations, to methods which consider low-level appearance and motion cues. Recently, three general low-level representation schemes have been central in action recognition systems. These are the local descriptors (e.g., [13]), optic flow based methods such as [1], and dynamic-texture representations [8].

This paper does not present new appearance descriptors, rather we employ code made available by the respective authors of various existing contributions, specifically MIP [8], dogMIP and histMIP [5], and dense MBH [22]. The Motion Interchange Patterns (MIP) descriptor is a dynamic-texture representation, which reflects the range of possible changes in motion and their likelihoods of occurring at each pixel in the video. This is done by comparing 3×3 gray-level patches, for 64 different direction changes from a previous frame, through the current frame, to the next frame. Each comparison provides a trinary value indicating the similarity of the central patch to the one in the previous frame or to the next, producing a characterization of the motion in the pixel without computing flow. Various combinations of MIP trinary values indicate static edges which may be ignored by subsequent processing. In addition, MIP codes allow for effective camera motion compensation. Since originally presented in [8], several variations of the original MIP descriptor have been proposed and have gradually improved its capabilities [27, 5].

The second type of descriptor used in our experiments, is the dense-motion boundary histogram (dense-MBH) descriptor proposed by [22]. It uses dense, random sampling of multi-scale part models as an effective, yet efficient, action representation. Each of these "parts" is a fixed-dimensional cube in space-time, which locally captures both the spatial and temporal appearance of the video. These parts are arranged in a pyramid which contains both a coarse-level sample of the video at half of its resolution, as well as multiple fine-scale cubes, overlapping in both space and time, sampled in locations determined by the position of the coarse-scale sample. The parts are sampled from an MBH representation of the video, originally described in [2]. This representation encodes the changes in optical flow along each image axis.

The field of action recognition is rapidly changing, and the descriptors we employ, although state-of-the-art when published (both in the last two years), are no longer leading the performance charts. Specifically, the most successful method to date is based on keypoint trajectories and combining multiple descriptors [24, 25], as detailed below.

2.2. Spatio-temporal (mid-level) structures

Local appearance and motion representations, and the associated Bag of Features models, are popular due to their simplicity, robustness and the ability to quickly train these models on large amounts of data. However, many authors have opted to utilize mid-level structures that are constructed hierarchically on top of the local features. These attempts have often helped push the performance envelope. Some examples include "Action Bank" [21], which employs spatio-temporal pooling of features; the Hierarchy of Discriminative Space-Time Neighborhood Features [11], which groups together local features to form discriminative structures; the stacked convolutional network approach of [15], which builds a hierarchy of spatio-temporal units; and the data mining approach of [4]. It is noteworthy that each of these examples used a technique that was very different than the others – such variance might imply a rapid evolution of methods.

A successful thread of work in action recognition considers dense video trajectories [24, 25, 19]. Feature points in video are tracked and their appearance descriptors accumulated along the trajectory to form complex descriptors. The properties of the motion trajectories themselves are also informative, and are encoded as normalized displacement vectors. Combining these sources of information, such approaches dominate the performance charts of most of the important action recognition benchmarks.

Our proposed method is designed to capture spatial structures, but could easily be extended to capture temporal relations as well. While it is intuitively appealing to think of video as a 3D entity, we first focus on the spatial structures. In addition, instead of capturing well localized structures, which is also an appealing idea with useful outcomes, we encode globally using representations that are learned locally, but never made explicit. This is done through the use of a 2D variant of the word2vec method.

2.3. word2vec

In a scheme that is much simpler than previous work in Natural Language Processing, where neural networks with many hidden units and several non-linear layers were con-

structed, word2vec [16] constructs a simple log-linear classification network. In the CBOW variant of word2vec, the network predicts each word based on its neighborhood – the five words preceding and the five words following that word. An input layer denotes the bag of words representation of the surrounding words, and contains one input element per each dictionary word. It is projected linearly to the hidden encoding layer. The hidden layer is then mapped to an output Huffman code representation of the given word. Once the network is trained, the projections from each input unit to the middle encoding layer are used to represent the associated dictionary word. Interestingly, the resulting encoding not only captures meaningful word representations, where words of similar meaning have nearby representations, but also captures, in a surprising manner, pairwise relations through simple arithmetic operations [16].

2.4. Bag of visual keywords variants

Bag of visual words approaches [23] represent a video V by a set of vectors $Q = \{I_i\}_{i=1}^{n}$. Each vector I_i encodes the local appearance of one video patch. The basic scheme for transforming the set Q into a vector is borrowed from the field of text processing. Using a dictionary of prototypes $D = \{C_j\}_{j=1}^{f}$, a histogram is constructed where each local descriptor is assigned to one bin according to its closest dictionary word $C_{i(j)}$.

In more sophisticated schemes, used for image recognition, such as sparse coding [26, 29], each keypoint is assigned with unequal weights to multiple dictionary prototypes, again based on its appearance. In very successful recent methods, including the Fisher Vectors [18], and VLAD [7], the appearance vectors themselves (e.g., 128-D for SIFT) are accumulated for each learned prototype.

Fisher vectors are based on a Gaussian Mixture Model (GMM) representation of the space of image descriptors. Assuming that the GMM has f centers $\mu_j, j = 1..f$, the representation of an image is a concatenation of f vectors. Each such vector is a weighted sum of the image descriptors I_i, where the weights are a function of the affinity between each image descriptor I_i and each GMM center μ_j. Somewhat similarly, in the VLAD method, a dictionary of some size f is used to generate a descriptor, which is formed by the concatenation of f vectors. Each one of these vectors $j = 1..f$, is the sum of vectors of the form $I_i - C_j$, for all descriptors of index i which are assigned to dictionary item j. The utility of the Fisher Vectors for action recognition was recently demonstrated in [25].

What is common to these schemes is that once assigned, the keypoint descriptors are considered as either atomic units or otherwise as vectors in the appearance space. There is an opportunity to utilize additional information by separating the two steps: (1) mapping image/video keypoints to prototypes based on appearance; (2) using accumulating schemes such as VLAD or Fisher Vectors but based on vector representations that are *not necessarily* related to appearance. This, in contrast to previous methods which couple both these steps together, and use appearance for both prototype assignment *and* accumulation of vectors.

3. The Piggyback Descriptor

3.1. Overview

Space-time keypoints are often considered atomic units, which can be counted, in parallel, by their type. However, in action recognition in particular, there is a line of work which tries to combine co-occurring keypoints into more elaborate structures. We suggest representing each keypoint as a vector, capturing not its appearance, but rather its spatial relation to other keypoints. In order to capture the spatial structures of keypoints, relating to the distribution of various types of keypoints within a single frame, we propose an encoding that is based on keypoint co-occurrences as represented by a 2D word2vec-like architecture.

Given all the keypoints of the training frames, we consider for each such keypoint a spatial neighborhood containing its $N = 10$ closest keypoints in the same frame. Recall that each keypoint i is assigned to a dictionary word $j(i)$. This assignment, similar to bag of keyword methods, is based on the similarities of the appearance representations of these keypoints. Each neighborhood is represented as a binary vector of the same size as the dictionary (f), which denotes exist/does not exist for each dictionary word. A network analogous to the word2vec network is trained to predict from this binary vector the type $j(i)$ of keyword i; that is, to predict the dictionary word association of each keypoint, based on its spatial neighborhood. This network has one hidden layer of size L, which is fixed to be 200 throughout our experiments. Once trained, we take the projection weights mapping the corresponding element in the input layer to the hidden units, as our encoding of each dictionary word $j = 1...f$. Hence, each word in our modified dictionary now reflects not its appearance, but rather the characteristics of the spatial arrangement patterns found in the video, in the vicinity of that word.

Representing an entire video is performed in a process similar to the one used by VLAD: The dictionary words are used to form a Gaussian Mixture Model (GMM) in $\mathbb{R}^L$ of $k = 5$ centers. Each dictionary word j is assigned to a Gaussian (based on the posterior probability) and its difference from the mean of that Gaussian is stored as a vector $U_j \in \mathbb{R}^L$.

Given a video to encode, the system produces a concatenation of k accumulator vectors in $\mathbb{R}^L$, where each of the keypoints adds to one of the k vectors the associated vector U_j. Thus, association of words to Gaussians, and their accumulation, is performed using one representation, "pig-

gybacking" on the original representation, which was used to form the original associations of features to dictionary words. The latter captures the appearance, the former, of size kL, is our contribution and is used to reflect spatial patterns.

3.2. Training

There are four components which are learned or computed: (i) a dictionary of keypoints containing f elements; (ii) Huffmann codes of the dictionary items; (iii) a representation of each dictionary element $1..f$ by a "semantic" vector in $\mathbb{R}^{200}$; and (iv) an association of each semantic vector to one of k Gaussians, and the center $\mu_l, l = 1..k$ of these Gaussians. These are mostly straightforward and are described below to allow complete reproducibility..

To learn the dictionary, we simply examine the set of training videos, extract the keypoints' descriptors I_i from all frames of these videos, and employ $k-means$ with the number of centers set to $f = 5000$ for MIP and its variants, and $f = 4000$ for dense MBH. Each training keypoint I_i is then associated with the closest cluster center $C_{j(i)}$.

The Huffman codes $H_j, j = 1..f$ of length $log_2(f)$ are constructed by considering the frequency of training keypoints associated with each dictionary item C_j. The use of Huffman codes follows a common practice in neural net language models [17]. We adhere to this practice, although its main advantage over encoding through balanced trees is to reduce evaluation time. This consideration is not a major concern in our system, where f is relatively small.

In order to obtain the semantic representation, we apply a variant of word2vec, as illustrated in Figure 1. Recall that each keypoint I_i is associated with the nearest dictionary item $C_{j(i)}$. The distances used for this association are based on the appearance descriptors, which are not used again in our procedure. The word2vec neural net is trained as follows: For each keypoint i in every training frame, we consider two vectors – a binary input vector and a binary output vector. The binary input vector B_i^{in} is of length f, with one element corresponding to each dictionary item. It denotes the existence of keywords assigned to each dictionary item within the neighborhood of size $N = 10$ surrounding keypoint i. The binary output vector $B_i^{out} = H_{j(i)}$ is the Huffman code of the associated dictionary item.

The neural network is trained to link the binary input to the binary output vectors using a simple log-linear architecture with a hidden (linear-projection) layer of size L. The model thus has a total of $f * L + L * log_2(f)$ parameters. Our implementation is based on the code available from https://code.google.com/p/word2vec/, and the learning procedure follows a gradient descent method with an adaptive learning rate controlled by the Adagrad scheme [3]. Once the network is trained, the semantic vector representation S_j of each dictionary item j is obtained as the projection weights from the associated input unit to the hidden layer.

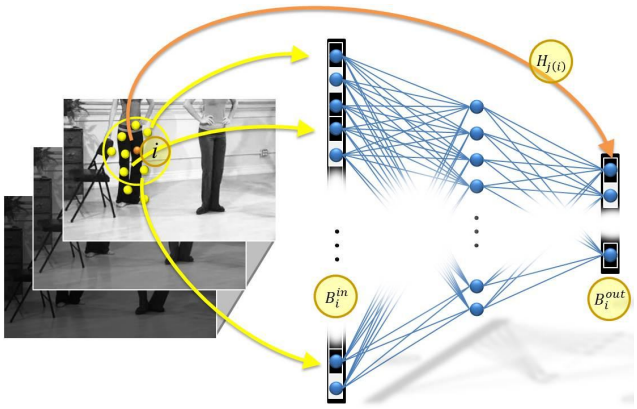

Figure 1: An illustration of the word2vec network applied to image or video descriptors. Each visual keypoint i is predicted, i.e, encoded at the output layer B_i^{out} with accordance to its Huffman encoding $H_{j(i)}$ of the closest dictionary item $j(i)$. The neighborhood of keypoint i is encoded through a bag of words scheme as the input layer B_i^{in}. The hidden layer is a linear projection layer and the entire architecture is a simple log-linear one.

We learn a mixture of $k = 5$ Gaussians in $\mathbb{R}^L$ by using all vectors $S_j, j = 1..f$ as input to an EM based GMM procedure. Once the GMM is trained, each dictionary item of index j is associated with the Gaussian of the highest posterior probability $a(j) \in 1, 2, ..., k$, and is represented by a vector $U_j = S_j - \mu_{a(j)}$, where μ_l is the center of the mixture's Gaussian of index l.

3.3. Encoding

Encoding a video, after the training completes, follows an efficient procedure. Keypoints are associated to vectors U_j and mixture index $1..k$ using the same flow used for training: each keypoint $i = 1..n$ in the video is represented by an appearance descriptor I_i and then by its closest (Euclidean distance) dictionary item $C_{j(i)}$. The dictionary item is associated with Gaussian number $a(j(i))$ and with descriptor $U_{j(i)}$ that was computed during training based on the semantic vector $S_{j(i)}$.

Let $A_q \subseteq 1..n$ be the set of all indices i for which $a(j(i)) = q$. The video is represented as the concatenation of k accumulator vectors $V_q, q = 1...k$ in $\mathbb{R}^L$, which are computed as:

$$V_q = \sum_{i \in A_q} U_{j(i)}$$

More elaborate representation schemes can be devised by splitting the video and using spatial and temporal pyramid representations. However, this is not explored here.

Table 1: Performance comparison of various combinations of methods on the ASLAN [10] benchmark. The average accuracy and standard error over the ten folds are given for a list of methods including MIP [8], histMIP and dogMIP [5], and MBH [2]. The MIP results are obtained from [8] and patchMIP and dogMIP results are taken from [5]. The Piggyback descriptor performs slightly worse than the baseline raw descriptor on which it is based. However, the combination, using stacking, of the baseline features and their Piggyback counterpart consistently improves results. The result of combining all descriptors (vanilla and Piggyback versions) together presents a leap in performance on the ASLAN benchmark.

System	Baseline		Piggyback		Combined	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
MIP	$64.62 \pm 0.8\%$	70.40	$63.50 \pm 0.6\%$	68.68	$69.70 \pm 0.9\%$	76.13
histMIP	$64.30 \pm 1.0\%$	69.09	$61.98 \pm 0.2\%$	66.38	$67.08 \pm 0.9\%$	73.61
dogMIP	$60.82 \pm 1.1\%$	65.76	$59.23 \pm 0.2\%$	63.29	$64.70 \pm 0.9\%$	69.71
MBH	$64.25 \pm 0.9\%$	69.91	$57.83 \pm 0.4\%$	61.39	$66.22 \pm 1.0\%$	72.18
All Combined	$65.88 \pm 0.9\%$	72.72	$64.47 \pm 0.4\%$	70.55	$72.25 \pm 1.1\%$	78.74

4. Experiments

We demonstrate our method on two challenging action recognition data sets: ASLAN [10, 9] and HMDB51 [12]. There are other popular action recognition benchmarks, see [6] for a comprehensive survey. Since we employ existing code as the underlying visual appearance descriptors, we choose in our experiments to focus on benchmarks for which publicly available code reproduces near state of the art results.

Classification is done using SVM. For ASLAN we use linear SVM following [8]; for HMDB the intersection kernel as in [22]. To combine methods, following [5], we use stacking on the signed distances from the hyperplanes of the individual SVM, and train a linear SVM on the entire training set. Almost all parameter values are taken from previous work. C=1 for ASLAN following [8, 5] and C=32.5 for HMDB following [22]. Dictionary sizes are also based on these baseline methods. The hidden layer has 200 units – the default value in [16]. The only "new" parameter is the number of GMM components, which we set to 5 in order to keep the representation compact. Experiments show that using one component (almost equivalent to summing the learned representations) does not work as well (1-2% lower accuracy on Aslan); we did not try other values so far.

The ASLAN benchmark contains $3,697$ videos from 432 different action classes. We follow the prescribed protocol that offers ten splits, each containing 600 pairs of video sequences. Half the videos are labeled as "same" and half as "not-same". At each round of the benchmark, nine splits are used for training, and one for testing. On the ASLAN benchmark, we apply the MIP descriptor [8], two recent variants of MIP [5], and MBH [2]. In addition, we apply the corresponding Piggyback descriptors. Experiments are conducted according to the common experimental settings involving multiple repeats of same/not-same trials.

Table 1 present the results on the ASLAN benchmark. For each of the MIP variants, and also for the MBH descriptor, the stacking of the baseline descriptor to its complementary Piggyback descriptor significantly outperforms baseline results. Combining representations, using the MIP or histMIP descriptors, already outperforms current state of the art result (AUC of 73.23), which was obtained in [5] by combining multiple descriptors. By combining all descriptors together, performance increases even further (AUC 78.74).

HMDB51 contains 51 different actions, with over a hundred samples each ($6,766$ sequences in total). Each of the three splits of the benchmarks contains 70 training videos per class. The rest are used for testing by way of multi-class classification. On HMDB51, although lagging behind the state of the art results, our results consistently show the complementary value of the Piggyback descriptor over the baseline dense MBH descriptor [22] (Table 2). The same pattern of results is common across each of the data sets. While the baseline appearance representation outperforms the suggested secondary "semantic" representation, their combination outperforms both.

We have also experimented with alternative schemes. For example, we replace the word2vec network with a Multidimensional Scaling (MDS) solution. Each dictionary item is represented as a point in $\mathbb{R}^L$ such that the underlying distances are optimized to match the distances between the associated binary input vectors B_i^{in} described in Section 3.2. Because each dictionary item has multiple occurrences in the video, and is associated each time with a different binary vector, this constitutes a generalization of MDS. In this generalization, every recovered MDS vector is repeated multiple times in the cost function. The optimization that is performed is a modification of the MDS EM algorithm [28]. The results are somewhat lower than those obtained with word2vec.

Since our method is accumulator-based, we also repeated the experiments on the ASLAN benchmark using the origi-

Table 2: Comparison of dense MBH [22] to its Piggyback version and to their combination. The average accuracy and standard error are shown. Dense MBH results were obtained using the authors' code and the results of the baseline system completely replicate their experiments. While the Piggyback version performs worse than the baseline method, combined they perform better than baseline. Since the publication of [22] earlier this year, much better results were obtained and the current state of the art on this benchmark is 57.2% [25].

System	Performance
Dense MBH	$43.51 \pm 0.5\%$
Piggyback of dense MBH	$31.22 \pm 0.7\%$
The two combined	$45.34 \pm 0.6\%$

nal VLAD descriptor [7]. Despite some effort, VLAD performed significantly worse than the baseline bag of word based methods and did not contribute to the overall performance even when used in combination.

References

[1] S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE TPAMI*, 32(2):288–303, 2010. 2

[2] N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In *Computer Vision–ECCV 2006, LNCS 3952*, pages 428–441. Springer, 2006. 2, 5

[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. 4

[4] A. Gilbert, J. Illingworth, and R. Bowden. Action recognition using mined hierarchical compound features. *IEEE TPAMI*, 33(5):883–897, 2011. 2

[5] Y. Hanani, N. Levy, and L. Wolf. Evaluating new variants of motion interchange patterns. In *Proc. IEEE Conf. Comput. Vision Pattern Recognition Workshops*, pages 263–268, 2013. 2, 5

[6] T. Hassner. A critical review of action recognition benchmarks. In *CVPR Workshops*, 2013. 5

[7] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, jun 2010. 1, 3, 6

[8] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, Oct. 2012. 2, 5

[9] O. Kliper-Gross, T. Hassner, and L. Wolf. One shot similarity metric learning for action recognition. *Proc. of the Workshop on Similarity-Based Pattern Recognition (SISM)*, pages 31–45, 2011. 5

[10] O. Kliper-Gross, T. Hassner, and L. Wolf. The action similarity labeling challenge. *IEEE TPAMI*, 34(3):615–621, 2012. 5

[11] A. Kovashka and K. Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *CVPR*, 2010. 2

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 5

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

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

[15] Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 2

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. 1, 3, 5

[17] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531, 2011. 4

[18] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, jun. 2007. 3

[19] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *CVPR*, pages 1242–1249, 2012. 2

[20] K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *MVAP*, Sept. 2012.

[21] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[22] F. Shi, E. M. Petriu, and R. Laganière. Sampling strategies for real-time action recognition. In *CVPR*, pages 2595–2602, 2013. 2, 5, 6

[23] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, Nice, France, 2003. 3

[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, 2013. 2

[25] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *International Conference on Computer Vision*, Oct. 2013. 2, 3, 6

[26] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360 –3367, jun. 2010. 3

[27] C. Whiten, R. Laganire, and G.-A. Bilodeau. Efficient action recognition with MoFREAK. In *Int. Conf. on Computer and Robot Vision*, pages 319–325, May 2013. 2

[28] S. Winsberg and G. Soete. A latent class approach to fitting the weighted euclidean model, clascal. *Psychometrika*, 58(2):315–330, 1993. 5

[29] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009. 3