Fast Non-parametric Action Recognition

Sebastián Ubalde and Norberto Adrián Goussies

Departamento de Computación, Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires, Buenos Aires, Argentina
{seubalde,ngoussie}@dc.uba.ar

Abstract. In this work we propose a method for action recognition which needs no intensive learning stage, and achieves state-of-the-art classification performance. Our work is based on a method presented in the context of image classification. Unlike that method, our approach is well-suited for working with large real-world problems, thanks to an efficient organization of the training data. We show results on the KTH and IXMAS datasets. On the challenging IXMAS dataset, the average running time is reduced by 50% when using our method.

Keywords: action recognition, nearest neighbor, image-to-class distance.

1 Introduction

The problem of automatically identifying an action performed in a video is receiving a great deal of attention in the computer vision community. The fields of application for action recognition are varied, and include video summarization, video indexing, video surveillance, human-computer interaction, etc. The problem is challenging because, on one hand, the appearance of an action can vary considerable in different videos and, on the other hand, different actions can look very similar to each other.

Several approaches to the problem have been proposed. An entire body of work [1–4] is based on a global representation of the video. Once the actor is localized in the video, movement information is encoded as a whole.

More related to our work are those approaches based on local descriptors. The basic idea is to characterize a video using descriptors of spatio-temporal patches extracted from certain interest points. A wide range of methods [5–9] have been used both for interest point detection and for descriptor computation. Many works use descriptor quantization in order to work with low-dimensional data. In [5, 6, 10–12], descriptors are clustered and cluster centers are selected as codewords. Videos are therefore represented as histograms of codewords. This approach is commonly known as *bag-of-features*. A classifier is trained using the set of histograms from the training videos. Nearest neighbor and support vector machines are among the most used.

Recently, Boiman et al. [13] proposed a method for image classification referred to as *Naive-Bayes Nearest-Neighbor* (*NBNN*), with several attractive features. First, it is a *non-parametric* classifier, which means that it needs no intensive learning phase. This is extremely useful when working with large training databases that are subject to frequent updates. Second, it achieves a performance comparable to that of the top *learning-based* methods. Learning based methods require an intensive parameter learning phase, and

L. Alvarez et al. (Eds.): CIARP 2012, LNCS 7441, pp. 268–275, 2012.

can usually achieve a better classification performance than non-parametric methods. According to [13], the success of the NBNN method is due to the avoidance of local descriptors quantization and to the use of *Image-to-Class* (I2C) distance instead of *Image-to-Image* (I2I) distance. As an extra advantage, the idea behind the method is fairly simple.

Despite its good qualities, the NBNN method is not well-suited for most real-world problems [14]. The number of training features required at those scenarios to achieve a state-of-the-art performance is usually very large. This makes I2C computation expensive and results in prohibitive classification times. In Sect. 3 we propose an alternative method to NBNN (named *NBNNTree*), by which we aim at lowering the amount of time consumed for classification.

Few studies ([14, 15]) have used NBNN for action recognition. In this work, we thoroughly tested NBNN and NBNNTree on two very popular action recognition datasets: the KTH dataset and IXMAS multiview dataset. To the best of our knowledge, this is the first time the NBNN approach is tested on the challenging IXMAS dataset.

2 The NBNN Method

While in the work of Boiman et al. NBNN is used for image classification, this paper deals with action recognition. Because of that, we use a slightly different terminology here, to reflect the fact that we are working with *videos* and *actions* instead of *images* and *classes*.

Let V be a query video, and let $d_1, d_2, \ldots, d_n$ be its local descriptors. The NBNN method chooses the action $\hat{A}$ performed in V according to the following equation:

$$\hat{A} = \underset{A}{\operatorname{argmin}} \sum_{i=1}^{n} \|d_i - NN_A(d_i)\|^2. \tag{1}$$

where $NN_A(d_i)$ is the nearest neighbor of d_i within the descriptors of action A. Descriptors of action A are gathered from every training video labeled with A. As Boiman et al. show [13], the summation in (1) approximates a *Video-to-Action* (V2A) KL-distance. In other words, NBNN computes an approximated distance from V to every possible action, and chooses the action with the minimum distance.

As shown in [14], NBNN requires a large number of local descriptors in the training set to achieve state of the art performance. This makes the computation of $NN_A(d_i)$ in (1) very expensive for real-world sets (which are usually built extracting more than 10000 descriptors per training instance). This is the main computational bottleneck, even when approximate searches (using KD-trees as in [13]) are performed.

Based oh the previous observation, it seems reasonable to expect that a reduction in the number of NN searches would lead to a more time efficient method. A first step in this direction is to notice the sequential nature of the NBNN method. Only after computing the V2A distance for every action, the method chooses the closest action. This may seem like a fair strategy, but it doesn't take advantage of a very common phenomena in action recognition problems. In most of them, actions can be easily arranged in sets of look-alike actions, each set containing actions similar to each other but not similar to actions in other sets.

For example, in the KTH dataset [16] two sets are distinguishable at first glance: the one consisting of actions *boxing*, *hand waving* and *hand clapping* and the one consisting of actions *running*, *jogging* and *walking*. It would take a very bad classifier to classify a *running* video as belonging to any action in the first set, or a *boxing* video as belonging to any action in the second set. Taking this into account, it seems inefficient to compute the V2A distance for every action. It would be much more efficient to quickly discard the wrong set, concentrating the efforts in choosing an action within the right set. This is precisely the idea behind our proposed method.

3 The NBNNTree Method

Our method is based on a particular organization of the descriptors in the training dataset. Instead of grouping descriptors according to their action (as in NBNN), we group them according to their *action-set*. An action-set is just a set of actions (e.g. the set $\{boxing, \ hand \ waving, hand \ clapping\}$).

The method requires a training step in which all actions are organized in an *action-set tree*. An action-set tree is a binary tree in which every subtree is labeled at its root with an action-set. For the purposes of our method, we are interested only in those action-set trees which are *valid*. A valid action-set tree t can be described as follows. If t is a leaf, then it should be labeled with an action-set consisting of a single action. If t is not a leaf, then the following conditions should be met:

1. Let s be the action-set label of t. Let s_l and s_r be the action-set labels of t's left and right subtree respectively. Then, $\{s_l, s_r\}$ should be a partition of s, with $|s_l| = \lfloor \frac{|s|}{2} \rfloor$ and $|s_r| = \lceil \frac{|s|}{2} \rceil$.
2. The left subtree of s should be a valid action-set tree.
3. The right subtree of s should be a valid action-set tree.

Figure 1 shows two examples of action-set trees. Every action-set in the picture is tagged with a letter for future reference. Figure 1 (b) shows an invalid tree. The tree is invalid for several reasons. First, $\{B, C\}$ is not a partition of A. Also, $|C| \neq \lceil |A|/2 \rceil$. Second, $\{D, E\}$ is not a partition of C. And third, B is at a leaf, but $|B| > 1$.

Based on the action-set tree, a *descriptor tree* is built. A descriptor tree is just an action-set tree in which every subtree stores at its root descriptors of the actions in its associated action-set. Descriptors are selected randomly from the training descriptor dataset. The exact number of descriptors q to select is chosen based on the confusion matrix obtained from our NBNN tests. Let s, s_l and s_r be defined as before. Let z be the number of descriptors per action in the training dataset, and m the confusion matrix for NBNN. The degree of confusion between actions in s_l and actions in s_r is given by $d(s_l, s_r, m) = \sum_{(a,b) \in (s_l \times s_r)} m(a, b) + m(b, a)$. To compute q, we first normalize $d(s_l, s_r, m)$, dividing it by $2|s_l||s_r|\alpha$, where α is just a reasonable confusion value between actions (we used 1% for all our experiments). Whenever q results larger than z, we simply discard it and use z instead.

Given a query video, both its local descriptors and the descriptor tree are used by the NBNNTree method for classification. The algorithm for the NBNNTree classifier is detailed in Algorithm 1. We use $left(t)$ and $right(t)$ to designate the left and right

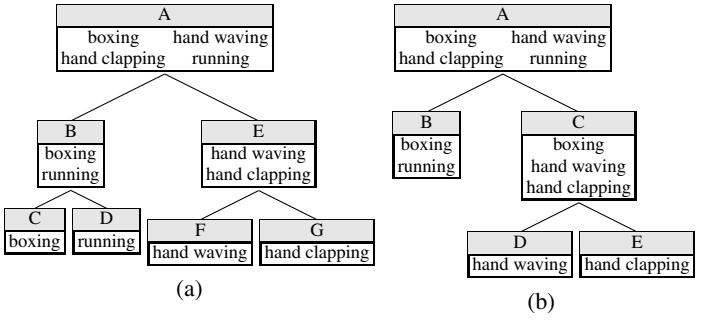


Fig. 1. Examples of valid and invalid action-set trees. Figure 1b shows an invalid tree. Figure 1a shows a valid tree.

subtrees of t respectively, $action\text{-}set(t)$ to designate the label at the root of t and $descriptors(t)$ to designate the set of descriptors stored at the root of t.

$NBNNTree(d_1, \ldots, d_n, t)$

Returns an action for a given set of descriptors $d_1, \ldots, d_n$ and a descriptor tree t

Steps:

1. If t is a leaf, return the only action in $action\text{-}set(t)$.
2. Let $L = descriptors(left(t))$ and $R = descriptors(right(t))$.
3. $\forall d_i$ compute the nearest neighbor of d_i in L: $NN_L(d_i)$.
4. Compute $D_L = \sum_{i=1}^{n} \|d_i - NN_L(d_i)\|^2$.
5. $\forall d_i$ compute the nearest neighbor of d_i in R: $NN_R(d_i)$.
6. Compute $D_R = \sum_{i=1}^{n} \|d_i - NN_R(d_i)\|^2$.
7. If $D_L < D_R$, $t_{next} = left(t)$, else $t_{next} = right(t)$.
8. Recursively call $NBNNTree(d_1, \ldots, d_n, t_{next})$.

Algorithm 1. NBNNTree algorithm

The algorithm starts from the root of the tree and descends one level at a time. At each level, the distance D_L between the video and the action-set in the left subtree is compared to the distance D_R between the video and the action-set in the right subtree. The algorithm descends to the subtree with smaller distance and the process is repeated. When a leaf is reached, its action-set (consisting of a single action) is returned.

Our method avoids the sequential strategy of the NBNN method. This was motivated by the observation that actions can be arranged into increasingly smaller sets of look-alike actions, yielding an action-set tree. But the question remains of how to build that tree. At this work we use a recursive method based on the confusion matrix from our NBNN tests. The method is shown in Algorithm 2.

$$c(s_l, s_r, m) = \sum_{a,b \in s_l \wedge a \neq b} m(a,b) + m(b,a) \quad + \sum_{a,b \in s_r \wedge a \neq b} m(a,b) + m(b,a) \qquad (2)$$

$BuildTree(s, m)$

Builds an action-set tree t from a given set of actions s and a given confusion matrix m

Steps:

1. $action\text{-}set(t) = s$.
2. If $|s| = 1$ return.
3. Split s into two subsets s_l and s_r, such that $c(s_l, s_r, m)$ (see (2)) is maximized, with $|s_l| = \lfloor \frac{|s|}{2} \rfloor$ and $|s_r| = \lceil \frac{|s|}{2} \rceil$.
4. $left(t) = BuildTree(s_l, m)$, $right(t) = BuildTree(s_r, m)$.

Algorithm 2. Building the action-set tree

At each step, the algorithm evaluates every possible partition of the set of actions s into a pair of sets s_l and s_r. The chosen partition is the one that maximizes $c(s_l, s_r, m)$. As shown in (2), $c(s_l, s_r, m)$ is the sum of confusion values between actions in s_l plus the sum of confusion values between actions in s_r. Thus, s_l and s_r are chosen in such a way that actions belonging to the same set are often confused with each other by the NBNN classifier.

3.1 Time Complexity of the NBNNTree Method

As shown in the previous section, NBNNTree performs several steps at each level of the tree (numbered from 1 to 8 in Algorithm 1). Of these steps, 1, 2, 7 and 8 are clearly $\mathcal{O}(1)$. Step 3 computes $NN_L(d_i)$ for each of the n descriptors of the query video. Following [13], we used an approximate nearest neighbor algorithm [17] for the computation of $NN_L(d_i)$. The expected time for this nearest neighbor search is logarithmic in $|L|$. Thus, step 3 is $\mathcal{O}(n \log(|L|))$. Step 4 is clearly $\mathcal{O}(n)$, because it involves only $\mathcal{O}(1)$ computations over n values. Step 3 to 4 together are therefore $\mathcal{O}(n \log(|L|))$. Likewise, steps 5 to 6 are $\mathcal{O}(n \log(|R|))$. In our experiments, every node in the descriptor tree stores at most z descriptors, where z is the number of training descriptors for a single action. Replacing $|L|$ and $|R|$ for z in the previous expressions yields a time complexity of $\mathcal{O}(n \log(z))$ for the steps $1 - 8$ performed at each level of the tree. Since t is built from a valid action-set tree, its height is logarithmic in the total number of actions k. Thus, the time complexity of the NBNNTree classifier is $\mathcal{O}(\log(k) n \log(z))$. This is a substantial speed-up over the NBNN method, which has a time complexity of $\mathcal{O}(kn \log(z))$.

4 Experiments

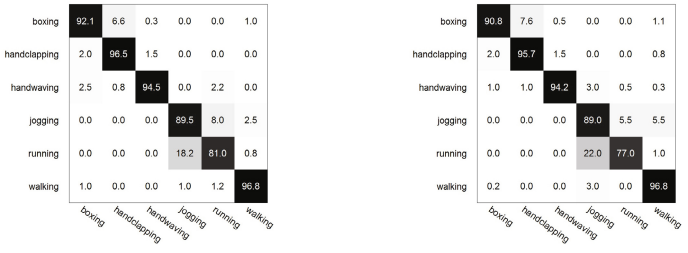
We tested NBNN and NBNNTree on two well-known action recognition datasets: the KTH dataset [16] and the IXMAS dataset [18]. For all the experiments, descriptor datasets were built following [6]. Parameters for NBNN were set as suggested in [13].

The KTH dataset contains 6 actions, performed several times by 25 actors in 4 different scenarios of appearance, illumination and scale. Both camera location and orientation of actors remain constant for most videos. In total, the dataset consists of 2391 videos. Figure 2 shows some example frames.

(a) (b) (c) (d)

Fig. 2. Example frames from KTH dataset: *walking* (a), *jogging* (b), *running* (c), *boxing* (d)

We used leave-one-out cross-validation (LOOCV) on the actors. That is, videos were divided into 25 sets, each including exactly the videos of one actor. In each of 25 experiments, the classifier was trained using the videos from 24 sets and tested on the videos from the remaining set. We report average precision (both global and by action) over the 25 experiments. Figure 3 shows confusion matrices for NBNN and NBNNTree. Our method obtains similar results to those of NBNN for all the actions. Table 1 (a) compares our results on KTH with those of existing approaches.



(a) (b)

Fig. 3. Confusion matrices for the KTH dataset using NBNN (a) and IXMAS (b)

The average running time for classification is reduced by 10% when using NBNNTree, compared to NBNN. This improvement came almost entirely from the strategy used to select the number of stored descriptors at each node, detailed in Sect. 3. For some of the nodes, only 8% of the total number of descriptors for that node action-set were enough to achieve the results reported in Fig. 3 (b), which is impressive. Because of the small number of actions of the KTH dataset, the main improvement in time complexity provided by NBNNTree is not fully exploited.

The IXMAS dataset contains 13 actions, performed 3 times by 12 actors. Each action execution was recorded by fixed cameras at 5 different positions. We use the videos of the 4 cameras usually considered in literature. Actors arbitrarily chose position and orientation. Figure 4 shows some example frames.

We used 6-fold cross-validation on the actors, and report average precision (both global and by action) over the 6 experiments. Classifiers were trained using the videos taken by the four cameras, and tested on the videos of one designated camera. Table 1 (b) compares the results achieved by several approaches. Both NBNN and NBNNTree perform better than the rest of the methods. Confusion matrices are shown in Fig. 5. Average running time for classification is reduced by 50% when using NBNNTree,

(a) (b) (c) (d)

Fig. 4. Example frames for the dataset IXMAS: *walk* (a), *wave* (b), *punch* (c), *kick* (d)

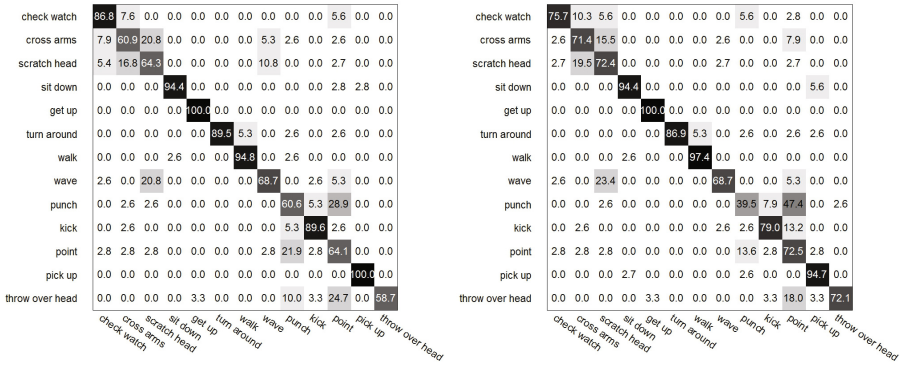
Table 1. Average precision (in %) of different methods. Table (a) shows results for the KTH dataset. Table (b) shows results for the IXMAS dataset.

(a)

Method	Precision
Dollar et al. [6]	80.66
Liu et al. [11]	94.16
Laptev et al. [5]	91.8
Nowozin et al. [19]	84.72
NBNN	91.73
NBNNTree	90.58

(b)

Method	Cam 0	Cam 1	Cam 2	Cam 3
Weinland et al. [2]	65.4	70	54.3	66
Yan et al. [3]	72	53	68	63
Liu et al. [11]	76.67	73.29	71.97	72.99
NBNN	79.4	76.15	74.04	74.13
NBNNTree	78.8	75.83	71.07	71.75



(a) (b)

Fig. 5. Confusion matrices for the IXMAS dataset, using NBNN (a) and NBNNTree (b)

compared to NBNN. Due to the larger number of actions, the benefits of using the descriptor tree are more clearly visible with IXMAS than with KTH.

5 Conclusions

We proposed a method that significantly reduces the time complexity of NBNN while achieving a similar classification accuracy. Our approach benefits from the good qualities of non-parametric methods, and is better suited for real world problems. On the challenging IXMAS dataset, the average running time is reduced by 50% when

using our method, compared to NBNN. In the future, we hope to explore new criterias to organize descriptors, in an effort to further improve classification time.

References

1. Davis, J., Bobick, A.: The representation and recognition of action using temporal templates. In: CVPR 1997, pp. 928–934 (1997)
2. Weinland, D., Boyer, E., Ronfard, R.: Action recognition from arbitrary views using 3d exemplars. In: ICCV 2007, pp. 1–7 (2007)
3. Yan, P., Khan, S., Shah, M.: Learning 4d action feature models for arbitrary view action recognition. In: CVPR 2008, pp. 1–7 (2008)
4. Zelnik-manor, L., Irani, M.: Event-based analysis of video. In: Proc. CVPR, pp. 123–130 (2001)
5. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR 2008, pp. 1–8 (2008)
6. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: PETS 2005, pp. 65–72 (2005)
7. Le, Q., Zou, W., Yeung, S., Ng, A.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: CVPR 2011, pp. 3361–3368 (2011)
8. Goussies, N.A., Liu, Z., Yuan, J.: Efficient search of top-k video subvolumes for multi-instance action detection. In: ICME 2010, pp. 328–333 (2010)
9. Yu, G., Goussies, N., Yuan, J., Liu, Z.: Fast action detection via discriminative random forest voting and top-k subvolume search. MultMed. 13(3), 507–517 (2011)
10. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV 2003, pp. 1470–1477 (2003)
11. Liu, J., Shah, M.: Learning human actions via information maximization. In: CVPR 2008, pp. 1–8 (2008)
12. Bregonzio, M., Gong, S., Xiang, T.: Recognising action as clouds of space-time interest points. In: CVPR 2009, pp. 1948–1955 (2009)
13. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: CVPR 2008, pp. 1–8 (2008)
14. Wang, Z., Hu, Y., Chia, L.: Learning instance-to-class distance for human action recognition. In: ICIP 2009, pp. 3545–3548 (2009)
15. Yuan, J., Liu, Z., Wu, Y.: Discriminative video pattern search for efficient action detection. PAMI 33, 1728–1743 (2011)
16. Laptev, I.: On space-time interest points. IJCV 64, 107–123 (2005)
17. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: International Conference on Computer Vision Theory and Application, VISSAPP 2009, pp. 331–340. INSTICC Press (2009)
18. Zelnik Manor, L., Irani, M., Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. CVIU 103, 249–257 (2006)
19. Nowozin, S., Bakir, G., Tsuda, K.: Discriminative subsequence mining for action classification. In: ICCV 2007, pp. 1–8 (2007)