Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Efficient descriptor tree growing for fast action recognition

S. Ubalde [*], N.A. Goussies, M.E. Mejail

Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina

ARTICLE INFO

Article history:
Available online 22 May 2013

Communicated by Luis Gomez Deniz

Keywords:
Action recognition
Nearest neighbor
Instance-to-Class distance

ABSTRACT

Video and image classification based on *Instance-to-Class* (I2C) distance attracted many recent studies, due to the good generalization capabilities it provides for non-parametric classifiers. In this work we propose a method for action recognition. Our approach needs no intensive learning stage, and its classification performance is comparable to the state-of-the-art. A smart organization of training data allows the classifier to achieve reasonable computation times when working with large training databases. An efficient method for organizing training data in such a way is proposed. We perform thorough experiments on two popular action recognition datasets: the KTH dataset and the IXMAS dataset, and we study the influence of one of the key parameters of the method on classification performance.

1. Introduction

Action recognition is a growing topic in computer vision research. Given a list of possible actions (e.g. running, sitting, clapping hands, etc.) and a video showing an actor performing any one of them, the goal is to produce an algorithm for the recognition of the action being performed. Automatic identification of actions in videos is a key aspect in many applications such as video summarization, video indexing, vigilance based on the analysis of security cam-captured videos, interaction with computers via movement, etc. The problem poses several challenges. On the one hand, the appearance of an action can vary considerably in different videos. This may be due to changes in lighting conditions, viewpoint, actor's clothing, etc. On the other hand, different actions can look very similar to each other. Occlusion and bad lighting conditions can add further difficulty to the problem.

Among recent works, those inspired on the *bag-of-features* approach became popular because of their simplicity and good performance. These studies base classification on measuring the distance between the query instance and each of the training instances – *Instance-to-Instance* (I2I) distance- and require quantizing instance features into a fixed-length vector for representation. Recently, Boiman et al. (2008) suggested that the use of I2I distance and feature quantization can severely hurt performance and proposed a method to overcome these issues. Their approach deals directly with unquantized features and computes *Instance-to-Class* (I2C) distances (instead of Instance-to-Instance) for classification. Apart from dealing with these problems, the method (referred to as *Naive–Bayes Nearest-Neighbor* or *NBNN*) presents 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 can usually achieve a better classification performance than non-parametric methods. 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 (Wang et al., 2009). 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 Ubalde and Goussies (2012) we proposed an alternative method to NBNN (named *NBNNTree*), by which we aimed at lowering the amount of time consumed for classification. Broadly, the strategy followed to achieve such improvement was that of organizing the training features in a particular fashion.

Our approach successfully reduced the time complexity of NBNN while achieving a similar classification accuracy. However, the training stage suggested in Ubalde and Goussies (2012) had a few loose ends, that limited its use to databases with a low number of actions. In Section 4, after a short introduction to our original method, we present a strategy to overcome these problems.

As far as we are aware, only (Wang et al., 2009; Yuan et al., 2011) have used NBNN for action recognition. In Section 5 we thoroughly test our improved version of the NBNNTree method on two very popular action recognition datasets: the KTH dataset and IXMAS multiview dataset. We compare its performance and computation times with those achieved by NBNN, and we investigate the influence of one of its main parameters on classification performance.

[*] Corresponding author. Tel.: +54 11 4541 1113; fax: +54 11 4576 3359.
E-mail addresses: seubalde@dc.uba.ar, sebastian.ubalde@gmail.com (S. Ubalde), ngoussies@dc.uba.ar (N.A. Goussies), marta@dc.uba.ar (M.E. Mejail).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.05.007

2. Previous work

Automatic analysis of actions and behaviors in video has been extensively analyzed in previous works. Existing approaches to address the problem are varied.

An entire body of work is based on a global representation of the video. Using tracking or background subtraction the actor is localized in the video, and movement information is encoded as a whole. The approach presented in Davis and Bobick (1997) is based on the so-called *temporal templates*. They extract silhouettes from several frames, and aggregate differences among them yielding two images that encode action information: motion history image (MHI) and motion energy image (MEI). Hu moments are used to compare two templates. Euclidean distance is used by Weinland et al. (2007) to match two silhouettes. More examples based on silhouettes can be found in Zhu et al. (2009), Souvenir and Babbs (2008) and Wang and Suter (2006). Several works use spatio-temporal volumes to represent an action. A spatio-temporal volume is formed by stacking frames over a given sequence. Examples of this approach can be found in Yilmaz and Shah (2008), Yan et al. (2008), Grundmann et al. (2008) and Zelnik-manor and Irani (2001).

More related to our work are those approaches based on local descriptors. Such approaches are derived from techniques used in image classification. The basic idea is to characterize a video using descriptors of spatio-temporal patches extracted from certain interest points. In the work of Laptev (2005), the Harris detector (Harris and Stephens, 1988) is extended to the space–time domain. Interest points are located using the extended detector. The resulting points can be thought of as spatiotemporal corners. Patches around them are expected to correspond to video objects whose movement is changing direction. Dollar et al. (2005) detect interest points using a Gaussian filter to the spatial dimension and a Gabor filter to the temporal dimension. Chomat et al. (2000) use the responses after applying spatio-temporal receptive fields. Rapantzikos et al. (2007) apply discrete wavelet transforms in each of the three directions of a video volume.

Patches around interest points are usually represented using descriptors. Descriptors are intended to provide distinctive information about the patch, while being invariant to appearance, occlusion, rotation and scale. In Laptev (2005) histograms of oriented flow and gradients are used as descriptors. Dollar et al. (2005) use image gradients and PCA to reduce descriptor dimensionality (Willems et al., 2008) use an extension of SURF features (Bay et al., 2006) to 3D.

Many works use descriptor quantization in order to work with low-dimensional data. In Dollar et al. (2005), Laptev et al. (2008), Sivic and Zisserman (2003), Niebles et al. (2006), Schuldt et al. (2004) and Liu and Shah (2008), 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 (NN) and support vector machines (SVM) are among the most used classifiers. While the first ones are easier to train, the last ones often achieve a better performance.

In Yuan et al. (2009) and Wang et al. (2009), descriptors are not clustered. Instead, they are used directly for classifying the video. This is based on the idea of Boiman et al. (2008) and presents several advantages over the bag-of-features approach, as discussed later in this paper.

3. 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 $\widehat{A}$ performed in V according to the following equation:

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

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 the summation in (1) approximates a *Video-to-Action* (V2A) KL-distance (Boiman et al., 2008). In other words, NBNN computes an approximated distance from V to every possible action, and chooses the action with the minimum distance.

3.1. NBNN drawbacks

As shown in Wang et al. (2009), 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 10,000 descriptors per training instance). This is the main computational bottleneck, even when approximate searches (using KD-trees as in Boiman et al. (2008)) are performed.

Based on 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 does not 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 (Laptev, 2005) 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.

4. 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: