# Action recognition in video

Pierre Kreitmann

**Abstract**

Automatic action recognition in video has a broad array of applications, from surveillance to interactive video games. Classic algorithms usually use hand-crafted descriptors such as SIFT (see [5]) or HOG (see [3]) to compute feature vectors of videos, and have achieved promising results in the past (see [7]). More recently, Quoc Le and Will Zou at the Stanford AI lab have proved that ISA features obtained from unsupervised learning achieve higher performance, while being much faster to engineer that hand-crafted features (their work is not yet published).

SFA features have achieved good results in object recognition as well as position and rotation extraction from artificial video signal (see [4]). In this work, we experiment using SFA features for action recognition.

## 1 Evaluation framework

Our pipeline for action recognition follows the pipeline presented in [7]. It first uses a dense detector to split the videos into spatio-temporal chunks. It computes the SFA feature vector of each of those chunks, and then clusters all those feature vectors (from all the videos). Finally, it defines the feature vector of one video to be the vector $(h_i)$, where $h_i = 1$ if and only if one chunk in the video is in the $i$-th cluster. Those feature vectors are then used by a SVM with a $\chi^2$ kernel for classification.

## 2 One layer SFA

We first experiment with a one-layer SFA. For all my experiments, we use movies from the Hollywood2 dataset (see [6]) at half the original resolution, ie. around $300 \times 150$ px.

The size of the patches is $10 \times 10$ px, so the input dimension is 100. A first SFA pass reduces the dimension to 48. Then, quadratic expansion is performed, and finally SFA selects the 32 slowest features, so that the output dimension is 32. Since the temporal size of each spatio-temporal chunk is 10 frames, the dimension of the feature vector of each spatio-temporal chunk is 320.

With the full dataset, the accuracy measured is 22.4 %. For comparison, a random guess would have an accuracy of 8.3 %. However, state-of-the art accuracy is around 50 % (47.4 % in [7], using HOG/HOF descriptors, but higher accuracy has been obtained with ISA by Quoc Le and Will Zou, though there work is not yet published).

Figure 1 shows that the accuracy doesn't increase much with the size of the training set. Indeed, using 400 movies gives approximately the same accuracy as using using 900 movies[1].

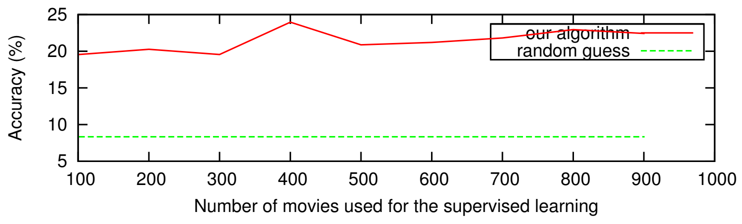

Figure 1: Accuracy as a function of the size of the training set

This suggests our relatively poor initial result is due to a high bias, which means our model doesn't capture enough information about the videos. Here are possible causes for this bias, that will be explored in the next subsections.

- The number of clusters is too low (see 2.1);
- The number of chunks is too low (see 2.2 and 2.4);
- The size of the chunks fail to capture important information (see 2.3).

## 2.1 Influence of the number of clusters

In the pipeline, the clustering process is used to compute the feature of each movie, and make sure they all have the same size. More precisely, the size of the feature vector is equal the number of clusters.

To determine the effect of the number of clusters $k$ on our results, we measure the accuracy obtained with different values of $k$, from 1000 to 4500. The other parameters are kept constant and have the same value as in our first measure.

Figure 2 shows that the accuracy globally increases until 3500 clusters, and then decreases. The best accuracy is obtained for $k = 3500$, and is worth 27.6 %.

## 2.2 Influence of the number of chunks per movie

In Hollywood2, each movie comprises about 10 000 spatio-temporal chunks (for $10 \times 10$ px spatial patches). Computing k-means with all the chunks from all the movies would require to store them all in memory. To avoid that, my version of the pipeline only uses a few samples from each movie (in the first experiment, we

[1]The algorithm is not deterministic: it uses random variables for selecting the features before k-means, which explains the irregularities in our graph.

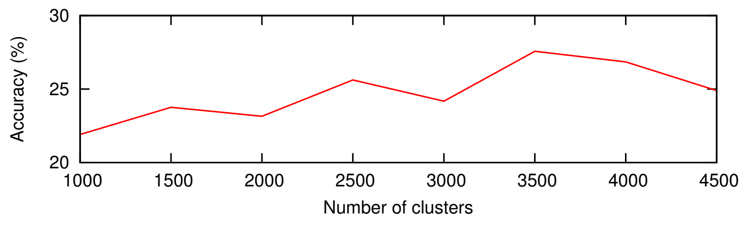

Figure 2: Accuracy as a function of the number of clusters

take 1000 random chunks from each movie), and compute k-means with those samples. Here, we want to determine if the number of chunks sampled from each movie has an influence on the accuracy.

Figure 3 shows that above 1000 chunks per movie, the number of samples doesn't have much influence over the result. It proves that our initial value (1000) was sufficient, and can't be the cause of our low accuracy.

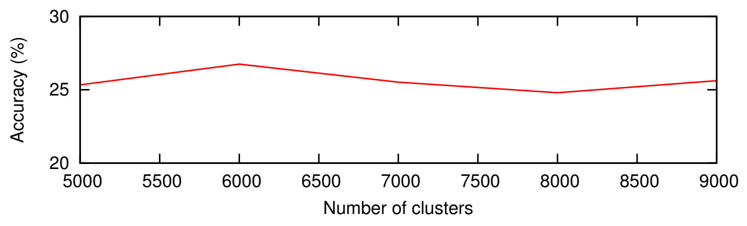

Figure 3: Accuracy as a function of the number of chunks sampled from each movie

## 2.3 Influence of the size of the patches

The computation of the feature vector of each movie relies on computing the SFA feature vectors of every spatio-temporal chunks in the movie. In our initial setting, the spatial dimensions were $10 \times 10$ px, and each frame of the chunk was converted into a 32-dimensional feature vector.

To investigate the effects of the spatial dimensions of the chunks on the accuracy, we test our algorithm with $15 \times 15$ px and $20 \times 20$ px, and keep the dimension of the feature vector constant (32). Table 1 shows that there is a small improvement when we use $15 \times 15$ patches, and that the accuracy then remains approximately constant for bigger patches.

## 2.4 Effect of spatial overlap

Previously, our dense detector returned adjacent patches in the image. Here, we investigate the effects of selecting patches with a 50 % overlap in the $x$ and

| Size | Accuracy |
|---|---|
| $10 \times 10$ px | 24.2 % |
| $15 \times 15$ px | 26.5 % |
| $20 \times 20$ px | 26.4 % |

Table 1: Accuracy as a function of the size of the patches

$y$ directions. This results in a finer grid, which contains 4 times more patches.

We compare two different settings: the first one uses the 50 % overlapping grid, and selects 4000 patches per movie. The second one doesn't use the overlapping grid, and also selects 4000 patches per movie (but out of 4 times less patches). Both experiments have 3500 clusters.

In the first case (with overlap), the accuracy is 25.9 %, while the second case (without overlap) yields an accuracy of 23.7%.

## 3 Two-layer SFA

Our one-layer SFA didn't produce very good results. We have tried several modifications, but only a few of them had a significant positive influence on the accuracy of our action detection algorithm: increasing the number of clusters and the size of the patch. Here, we'll investigate the use of a two-layer SFA detector instead.

### 3.1 Structure

As before, we use our first layer histogram, with patches of $10 \times 10$ px, and 3500 clusters. Then we feed a second layer with the output of the first layer. More precisely, each node of the second layer takes its input from 4 adjacent nodes of the fist layer. So, the input dimension of the second layer is $4 \times 32 = 128$. As in the first layer, a first pass of SFA reduces this dimension to 48. Then, quadratic expansion is performed, and finally the dimension is reduced to 32 by another pass of SFA.

Each output of the second layer thus represents a $20 \times 20$ px patch, and those patches overlap over 10 px, ie. 50 % in the two directions $x$ and $y$.

### 3.2 Results

The two-layer descriptor gives an accuracy of 25.6 %. This is not an improvement with regard to the first-layer descriptor.

## 4 Conclusion

We have applied SFA to action recognition in video, and our initial results were low (22.4 %). We have explored various possible ways to improve this accuracy, and have obtained 27.6 % with our best setting: a one-layer SFA, using 3500 centroids (see 2.1). This is still far from state-of-the-art performance. Finding ways to improve performance of SFA on natural images is an active research problem (see [2]).

## 5 Acknowledgements

My work for this project was supervised by Quoc Le and Will Zou, and I thank them for their guidance and their help, both for theoretical and technical problems.

My code uses the sfa-tk toolkit for matlab (see [1]) for training the SFA features, and I was helped by the code given by Quoc and Will.

## References

[1] P. Berkes. sfa-tk: Slow feature analysis tookit for matlab, 2003.

[2] Alistair Bray and Dominique Martinez. Kernel-based extraction of slow features: Complex cells learn disparity and translation invariance from natural images. In *In Advances in Neural Information Processing Systems*. MIT Press, 2002.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[4] Mathias Franzius, Niko Wilbert, and Laurenz Wiskott. Invariant object recognition with slow feature analysis. In Véra Kurková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks - ICANN 2008*, volume 5163 of *Lecture Notes in Computer Science*, pages 961–970. Springer Berlin / Heidelberg, 2008.

[5] David G. Lowe. Object recognition from local scale-invariant features, 1999.

[6] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context.

[7] Heng Wang, Muhammad Muneeb Ullah, Alexander Kläser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference*, page 127, sep 2009.