CS229 FINAL REPORT: UNSUPERVISED LEARNING OF TEMPORALLY COHERENT FEATURES FOR ACTION RECOGNITION

ZHIHENG HUANG, ROBIN JIA, AND KRIS SANKARAN

Introduction

Human vision is remarkably well-adapted to object and action recognition tasks. Recent literature suggests that at least some of the brains abilities to process visual cues can be explained by temporal coherence, the principle that signals closer together in time are more similar than those that are not. This idea arises from the observation that while models of vision used in machine learning often use still images as training data, humans learn to recognize objects based on a continuous stream of visual input. This continuous stream encodes not only the appearance of an object at a given moment in time, but also the frequencies of various types of transformations. Importantly, the frequency of a transformation correlates negatively with how important it is for object recognition; this fact may contribute to the formation of invariance in the human visual system [1]. In this project, we extend existing strategies for object recognition with temporal coherence to the task of action recognition in video files. We try to learn, in a completely unsupervised manner, filters that respond differently to different types of actions but are invariant to irrelevant transformations. We evaluate performance of our approach using the benchmark KTH action recognition data set, obtaining 89.05% classification accuracy. Further, we experiment with motion detection and interactive response map applications. We conclude this document with potential directions for developing this project.

Sparse Linear Auto-encoder Algorithm

To learn features in video data, we construct a sparse linear auto-encoder (SLA) corresponding to the first layer network outlined in [4], as shown in Figure 1. We train the network on small video cubes to obtain weights that connect visible units and hidden units. These weights can be viewed as filters that we can later use to generate features. When building these filters, we pool pairs of hidden units so that they form quadrature pairs; together, such pairs allow for superior features and invariance. For the t-th training example $x^{(t)}$, the response of the hidden units is given by $h^{(t)} = Wx^{(t)}$ and the response of the output units is given by $y^{(t)} = W^T Wx^{(t)}$, where W is the matrix of weights connecting visible units and hidden units. If H is our pooling matrix, then the activation of the pooling units is given by $p^{(t)} = \sqrt{H(Wx^{(t)})^2}$, where we apply the square-root and squaring operators element-wise. Given this formulation and N training examples, one way to construct a standard SLA is to minimize the objective function (1)

$$J = \sum_{t=1}^{N} \|x^{(t)} - W^T Wx^{(t)}\|_2^2 + \lambda \sum_{t=1}^{N} \|p^{(t)}\|_2 \tag{1}$$

where the first term minimizes reconstruction error and the second term promotes sparsity. However, we wish to learn features that use information regarding temporal coherence. In particular, we want

Date: 12/16/2011.

Acknowledgements: The authors are grateful for the guidance of Will Zou in giving us the idea of this project, theoretical instruction as well as his invaluable advice for us to identify problems and get the algorithm work.

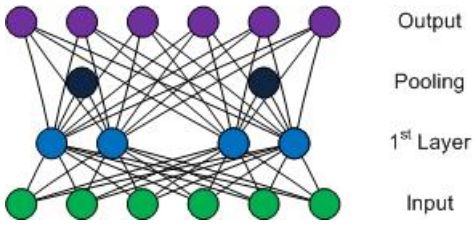


FIGURE 1. A graphical representation of our SLA. The links between the input and the hidden layer correspond to multiplication by W.

our hidden units to fire in a way that they retain the features of the data that change least over time: this way, they will be invariant to transformations of the data that disrupt the continuity of the scene being shown. Suppose that the N training examples are N sequential video blocks taken of some scene or object, perhaps in motion. We can concoct a new objective function that also penalizes differences in activation between sequential video blocks. More precisely, we let

$$J = \sum_{t=1}^{N} \|x^{(t)} - W^T W x^{(t)}\|_2^2 + \lambda \sum_{t=1}^{N} \|p^{(t)}\|_2 + \gamma \sum_{t=1}^{N-1} \|p^{(t)} - p^{(t+1)}\|_2. \tag{2}$$

We optimize this objective with LBFGS. We see that we wind up with edge detectors that pair up, due to pooling, as can be seen in Figure 2(a). An example visualization of the three-dimensional learned features, including moving edge-detectors, can be viewed at `http://stanford.edu/~kriss1/weightMatrices.mp4`. When preprocessing, we first scale the data to [0, 1], subtract out means and then whiten using PCA. We also tuned our parameters λ and γ to balance our emphasis on the reconstruction error, sparsity, and temporal coherence.

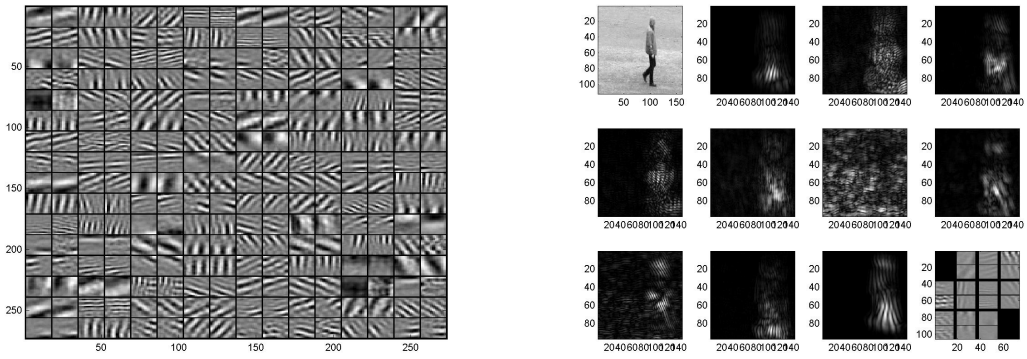
Visualizing Learned Filters and Generating Response Maps

Once we have generated the weight matrices for the hidden units in our auto-encoder, we can generate response maps via convolution. Each of the learned filters can be convolved with a video cube to generate a response map. When applied to a series of video frames, these response maps can then be used for motion detection or action recognition tasks. A screenshot of several response maps, along with their corresponding filters, is displayed in Figure 2(b). The complete video can be viewed online at `http://stanford.edu/~kriss1/walkingResponseMap.mp4`.

Experiments: Motion Detection and Interactive Response Map Generation

In addition to using our learned features to develop a classifier for action recognition, we experiment with using our features for motion detection and with generating response maps in real time. In the motion detection experiment, we first locate video blocks whose activation vectors have relatively large L_1 norms with non-overlapped sliding window. We then calculate a centroid for all these blocks, with their x, y coordinates weighted by L_1 norms of activation vectors; further, we adjust aspect ratio of the bounding box according to variances of the x and y coordinates. A clip of the motion detector applied to several KTH video clips is available at `http://stanford.edu/~kriss1/motion_detection_attention.mp4`. Note also that we have used motion detection to selectively dense sample motion-intensive areas in our action classification pipeline described below.

A second experiment was designed to interactively generate response maps. We implemented a system that takes a video stream from a webcam and uses an already-learned set of filters to create response maps in real-time. We thus have a means of applying our algorithm to our own video



(a) Visualization of filters learned using the SLA. (b) Screenshot from video of response maps.

FIGURE 2. Filter Visualization and Response Maps

data in real time, which could be useful for debugging and visualization purposes. A video of this application being used is available at `http://stanford.edu/~kriss1/interactiveMovie.mov`.

EXPERIMENTS: CLASSIFICATION ON BENCHMARK KTH DATA

To evaluate the effectiveness of this approach, we employed our learned features to classify actions in the KTH video data set–a standard benchmark test for action recognition algorithms. The KTH data set contains 2391 clips depicting 6 types of actions: running, walking, jogging, hand waving, hand clapping and boxing. 25 different people perform each of these actions, all against plain backgrounds. We follow the procedure proposed in [3] to obtain training and testing data. The classification pipeline we implemented includes five parts: data sampling, SLA feature extraction, K-means clustering of SLA featues, histogram feature extraction, and SVM classification using histogram features, as shown in Figure 3. In essence, we use a bag-of-words model to avoid processing the astronomical number of SLA features directly.

Sampling and SLA Feature Extraction. SLA feature vectors are formed by feeding video data to the SLA network and collecting activation values from all the pooling units. To obtain input data for the network, we employ a sliding window approach that takes relatively large stride to sample video cubes, as we have access to only limited computational resources. To further boost performance, we also developed a scheme that densely samples only in areas with the most significant SLA activation, i.e. interest points with activation vectors that have relatively large L_1-norm. In practice, this method has not brought significant improvement due to reasons that warrant further exploration.

SLA Feature Clustering and Histogramization. We use K-means clustering to discover centroids that may be representative of all extracted SLA features. As K-means can become trapped in local minima, we may run the algorithm several times with different initializations and use cross validation to pick a good set of centroids. The centroids are then used as the vocabulary for a bag of words model, where we bin all the SLA features from each video clip to create a histogram feature for it. Different schemes could be employed for the histogramization process. In our implementation, for each SLA feature, we increase the counting value of its corresponding bin by the feature's L_1-norm.

FIGURE 3. This is an overview of our classification pipeline for working with action recognition data.

Support Vector Machine Classification. The last step in our pipeline is to classify video clips using histogram features.The classifier we employ is libSVM, which essentially takes a one-versus-all scheme for multi-class classification. Following [4], we use chi-square kernel.

RESULTS

We are able to obtain an average of 89.05% accuracy in the KTH classification task using 5000 centroids for K-means clustering. The filters we used are trained on $16 \times 16 \times 10$ video cubes, with parameters $\lambda = 3$, $\gamma = 1$, 256 hidden units and pooling size 2. We apply PCA to reduce the dimension of input data to 512, retaining roughly 95% of variance. See Figure 2(a) for visualization of the filters. Also note that we do not train our filters on the KTH dataset but on a different dataset provided by Will Zou. We might examine in the future how training directly on KTH dataset will impact performance. In classification, we sample with a spatial stride of 8 and temporal stride 5. Generally using smaller stride would improve performance.

Table 1(a) compares our performance with published results using the same data set. Note that the state-of-art accuracy in [4] is 93.9%, which uses a similar SLA framework, but without temporal coherence. This does not mean that temporal coherence hurts performance, as [4] employs a second layer network, multi-scale sampling, and possibly more K-means iterations. We were not able to implement these features due to limited time and computational resources. Nevertheless, we compared our performance with a reduced version of [4] that omits the second layer and multi-scale sampling, has a vocabulary size of 3000, and uses one K-means iteration. We also retrained their filters using our training data. We repeated the the experiment for 3 times; the best and average accuracy can be found in Table 1(a). These results suggest that temporal coherence may help, although more convincing evidence would require beating state-of-art performance. Besides, Table 1(b) explores the impact of different parameter settings on performance. Lastly, We note that K-means sensitivity to local minima can have an approximately 2% influence on the result.

ERROR ANALYSIS

We analyzed the examples that our algorithm misclassifies. A large number of errors come from misclassifying similar-looking actions (e.g. hand waving, clapping, and boxing; jogging and running). In some cases, such as hand waving and hand clapping, the key to differentiating between actions depends on small details, such as whether the hands touch each other; in other cases, like running and jogging, the difference might be in how fast the person is moving. Our classifier also performs somewhat poorly when significantly different-looking videos belong in the same classification group. For example, our algorithm is good at recognizing that a person is walking if he/she is moving horizontally, but is prone to errors when the person walks in a diagonal direction.

FUTURE DIRECTIONS

SLA Training Data Selection. We hypothesize that we might be able to reduce error by selecting better training data for our SLA. Ideally, our data would be chosen to produce a balanced number of low frequency and high frequency filters; or we could train our SLA filter on the KTH dataset for this particular classification task.

TABLE 1. Performance

(a) Average accuracy by various algorithms on KTH dataset

Algorithm	Average Accuracy
Harris3D + HOG/HOF	91.8%
Harris3D + HOF	92.1%
Cuboids + HOG3D	90.0%
Dense + HOF	88.0%
Hessian + ESURF	81.4%
HMAX	91.7%
3D CNN	90.2%
pLSA	83.3%
GRBM	90.0%
Q. Le et. al. [4] Method with Dense Sampling	91.4%
Q. Le et. al. [4] Method with norm-thresholding	93.9%
Our Method (vocabulary size 5000)	89.05% (best 89.34%)
Our Method (vocabulary size 3000)	88.10% (best 88.52%)
Reduced Q. Le et. al. [4] Method (vocabulary size 3000)	85.40% (best 86.33%)

(b) Performance using different parameter settings

K-means iteration # = 1	
K-means centroid #	Average Accuracy
3000	88.10%
5000	89.05%
10000	88.53%

K-means centroid # = 3000	
K-means iteration #	Average Accuracy
1	88.10%
2	88.33%
3	88.18%

Multi-scale Classification. To better model the zoom-in and zoom-out effect as well as the changing size of moving objects in different clips, we might try to set up multi-scale sampling for our pipeline.

Second Layer Network. Following the example of [4], we would like to add a second layer to our network. We plan to construct a network similar to that of our sparse auto-encoder described above, but with some non-linearity. This could be achieved by utilizing the pooling output of first layer or using sigmoid function as activation function. Hopefully this layer would be able to aggregate information collected by the first layer filters and render a cumulative perspective on invariances in the data. In fact, we already have a preliminary implementation of this layer, though we have not yet had time to obtain reasonably-looking filters with it.

REFERENCES

[1] Einhauser, W., Hipp, J., Eggert, J, Korner, E., and Konig, P. "Learning viewpoint invariant object representations using a temporal coherence principle. *Biological Cybernetics* 93, No. 1 (May 2005): 79-90. `http://www.springerlink.org`

[2] Hateren, J.H. van and Schaaf, A. van der. "Independent Component Filters of Natural Images compared to the Primary Visual Cortex. *Proceedings: Biological Sciences* 265, No. 1394 (March 1998): 359-366.

[3] Wang, H., Ullah, M. M., Klser, A., Laptev, I., Schmid, C. "Evaluation of local spatio-temporal features for action recognition. *In BMVC 2010.*

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