Cortically Inspired Architectures for Action Recognition in Movie Clips

David Kamm, Jaehyun Park
Final Project Report for CS 379C
Stanford University, Spring 2009-10

June 9, 2010

1 Introduction

Video data is becoming a much more prevalent medium for transmitting information. While object classification in still images has been a standard task for the computer vision community, the importance of having computers understand video sequences is beginning to be recognized.

One task that has emerged from automated video understanding is action classification. In this task, the computer must correctly classify a video sequence which shows an action such as hugging or shaking hands. The original task was introduced in [4]. In their paper, they explored the correlation between action classification and scene classification. This relationship is not explored in our project as we are only concerned with action classification.

1.1 Motivation

Cortically inspired architectures have performed well on the task of object classification in still images on several standard data sets. However, it is not known how well architectures would do on the task of action classification, which requires temporal data. Our investigation is motivated by the fact that there is an increasing body of evidence that suggests that the cortex makes use of temporal information which these models do not account for. Such observations have led to methods such as Slow Feature Analysis (SFA), which try to capture features in scenes which do not vary over time [1].

A key component to cortically inspired architectures is the notion of invariance. According to [2], while *spatial* invariants have been thoroughly studied, not much work has been done towards investigating implementing models with *transformational* invariants. [2] defines a transformational invariant as one in which the dynamic structure stays the same while the spatial structure changes. An example would be different types of sports balls flying through the air. Even though the ball itself may vary in shape, a transformationally invariant system would recognize a ball moving through the air regardless. It would appear that having some form of transformational invariance would fit nicely in architectures designed for action recognition.

However, most cortically inspired architectures as described in [6] do not account for transformational invariance as described in [2]. Thus, the major question we tried to address was whether or not architectures like those described in [6] could perform well on the task of action classification. Additionally, we investigated the effect of adding a component to account for transformational invariance to such networks. This component, which does spatial-temporal pooling, is described in section 2.3.2.

1.2 Related Works

Recently, [6] investigated the impact of varying different factors in cortically inspired computer vision architectures in the context of single object recognition. Such factors included the number of hidden layers, the type of nonlinearity used for activating simple-cell like layer, and whether learning filters held any advantage over hardwiring them. Our project extends this work by investigating the impact of some of these variables in the context of action recognition with temporal data. At the time of this project, we have not encountered any published results on how well HMAX-like and other cortically inspired architectures perform on temporal datasets like *Hollywood2*.

Currently, the state-of-the-art result is by [5], which compared various local space-time features

for classification on the *Hollywood2* dataset. The model used in their study was a standard bag-of-features SVM approach for action recognition. They attained 47.4% of average accuracy across all action classes. They found that histograms of oriented gradients (HoG) and histograms of features (HoF) worked as the best descriptor for this dataset.

The Stanford AI lab has also started working on this action recognition task recently, but the project has just begun. Thus, another goal of our project is to provide some benchmark the performance of standard cortically inspired architectures to allow comparison with future work.

2 Detailed Implementation

2.1 Video Dataset

We used the *Hollywood2* dataset, which consists of 12 classes of human actions over 3,669 video clips [4]. The video clips are from 69 movies and vary in lengths. It is noteworthy that a video may be in multiple classes; for example, a video can contain both kissing and hugging. For our experiments, we extracted 40 frames per each video and scaled them down to 100×100 pixels grayscale images. If a video contains more than 40 frames, we picked 40 (not necessarily adjacent) frames from the middle of the videos. The frames are extracted such that they can cover as much portion of the video as possible but not too much to destroy temporal relationships between adjacent frames. This heuristic made the algorithms easier to implement.

2.2 Local Contrast Normalization

Before begin fed into the network, all video clips were normalized by two different techniques introduced in [6]. The subtractive normalization works as follows. We convolve each frame with an 8×8 Gaussian weighting window and subtracted the result from the original image. Divisive normalization is performed on the subtractively normalized data; it uses the same Gaussian weighting window and computes the "norm" for each area that it covers. Then, the subtractively normalized data is elementwise divided by these values. Formally, we had the following formulae:

$$v_{ij} = x_{ij} - \sum_{pq} w_{pq} \cdot x_{i+p,j+q} \quad (1)$$

$$y_{ij} = v_{ij} / \max\{c, \sigma_{ij}\} \quad (2)$$

$$\sigma_{ij} = (\sum_{pq} w_{pq} \cdot v^2_{i+p,j+q})^{1/2} \quad (3)$$

x represents the pixel values of the original frame. Also, v is subtractively normalized, and y is divisively normalized. w is a Gaussian weighting window such that $\sum_{pq} w_{pq} = 1$. For our experiments, the value of c is set to the mean of σ.

Figure 1 shows an example of this process for a single frame. Note that the normalization makes edges look more distinguished, especially the divisive normalization. Pictures are scaled for a better view.

2.3 Network Structure

In order to determine which structure works best for the dataset, we varied a number of different features of the network. The basic structure of our network is inspired by [7] and [8], and is feed-forward; no learning takes place in the model. Each layer of the network can be thought of as two sublayers: a simple cell-like layer and a complex cell-like layer. The simple cell-like layer achieves selectivity by applying filters to its input and then using a non-linear activation function on the weighted sum. The complex cell-like layer achieves invariance by pooling the activations of the simple cells over a small area.

2.3.1 Filters

We used different types of filters for simple cell layer as follows.

- Random filters
  Each pixel value of the filters are independently drawn from a normal distribution.
- Orthonormal random filters
  First, random filters are constructed. Then, the filters are orthogonalized such that the norm of each filter is 1.
- Gratings
  Regularly spaced parallel lines are used as filters, varying orientation and frequency.

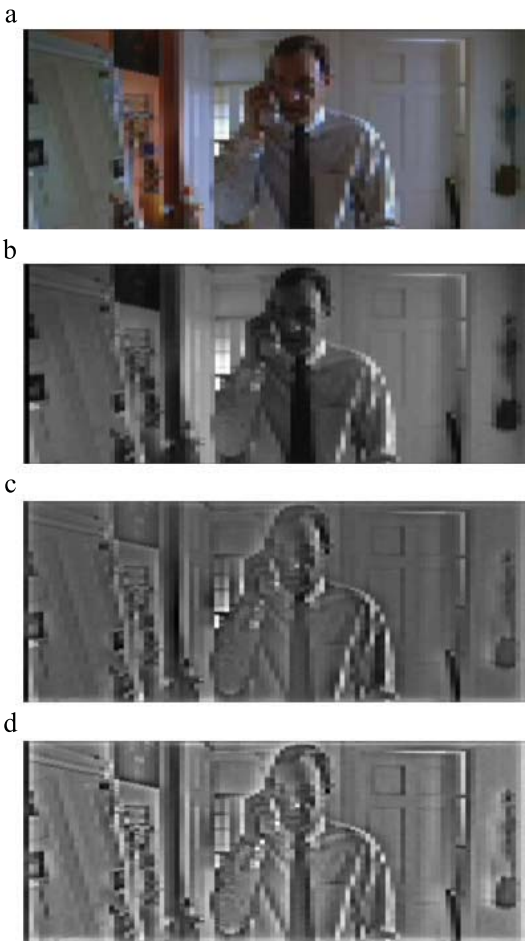

Figure 1: An example of normalization for a single frame. **(a)** The original frame. **(b)** The image converted to grayscale. **(c)** Subtractively normalized image. **(d)** Divisively normalized image.

- Gabor filters
  Each filter is a Gabor filter with different parameters: orientation, width, scale, wavelength, frequency.
- Gaussian filters
  Each filter is a Gaussian filter with different mean and covariance matrix. We also scaled and rotated the filters to make them ellipsoidal.
- Difference of Gaussian filters (DoG)
  Each filter is a Difference of Gaussian filter, which is a Gaussian filter subtracted from another filter having a lower variance.

2.3.2 Pooling Mechanism

Each complex cell pools over the activation from its receptive field to achieve invariance. The activation of a complex cell is determined by the following formula:

$$c = \sum_i \frac{s_i \cdot \exp(p \cdot |s_i|)}{\sum_k \exp(p \cdot |s_k|)} \tag{4}$$

Here, c is the activation of the complex cell and s_i represents the activation of the ith simple cell in the receptive field. The qualitative behavior of the function depends on p; if $p = 0$, c is the mean of s_i. As p approaches infinity, c approaches s_i having the maximum magnitude. For our experiment, we varied p from 0 to 10 to see its effect on classification accuracy.

In our architectures, we used two types of pooling. The first type of pooling was spatial only. In this, each complex cell pools over the activations of simple cells for one frame of the movie sequence. We can represent this efficiently by convolving a softmax kernel over each of the frames in the movie sequence to get the complex cell activations.

The second type of pooling we tried also factored in the temporal nature of the data. These architectures pool over the activations of retinatopically arranged simple cells over multiple frames. The activations are computed efficiently by convolving a "cube" kernel over the activations of the simple cells spanning multiple movie frames.

2.3.3 Number of Layers

As mentioned earlier, each layer in our model can be thought of as a pair of simple-complex cell sublayers. The simple cell-like layer employs a non-linear

activation on filtered inputs while the complex cell-like performs a softmax operation on its adherents from the simple cell-like layer below it. In our experiments we varied the number of simple-complex cell layers. Models either had one simple-complex cell layer or two.

2.3.4 Classification

For each video, activations of the cells in the highest layer were concatenated together to form a feature vector. Then, all feature vectors were passed to a support vector machine (SVM) to train a model and make predictions for test data. We ran the SVM for six different values for parameter λ from 10^{-3} to 10^2 to find the best result. We used 901 training examples and 972 test data.

3 Experimental Results

3.1 Local Contrast Normalization

To test the effect of local contrast normalization on classification accuracy, we passed raw pixel values as features into the SVM. Below is the result from this experiment.

Normalization	Accuracy
Grayscale	13.3%
Subtractive	16.3%
Divisive	**17.6%**

Table 1: Raw pixel classification accuracy using different normalization techniques

Another similar experiment was performed using activations from grating filters as features, as shown in Table 2.

Normalization	Accuracy
Grayscale	18.3%
Subtractive	**20.9%**
Divisive	20.8%

Table 2: Classification accuracy using different normalization techniques and grating filters

As it can be seen from above, local contrast normalization helped classification. Knowing that normalized data performs better, we also compared the the classification results using orthonormal random filters as below:

Normalization	Accuracy
Subtractive	19.1%
Divisive	**23.3%**

Table 3: Classification accuracy using different normalization techniques and orthonormal random filters

For the rest of our experiments, divisively normalized data was used.

3.2 Filters

We used different sets of filters for simple cells, using a single-layered network structure with softmax pooling. The result is shown in Table 4.

Filter	Accuracy
Raw pixel	17.6%
Random	22.5%
Orthonormal random	**23.3%**
Grating	20.8%
Gabor	21.4%
Gaussian	17.6%
DoG	14.8%

Table 4: Classification accuracy using different filters

3.3 Pooling Mechanism

To test the effect on the pooling mechanism, we tried different values for p for the softmax formula as the following. For the experiment, 120 orthonormal random filters were used. The result is shown in Table 5.

p	Accuracy
0	12.0%
1	12.9%
10	**14.3%**

Table 5: Classification accuracy using different pooling mechanism

3.4 Number of Layers

We also tried stacking another layer of simple cells and complex cells on top of the network, using the activations of the first layer's complex cells as inputs. For this experiment, we used Gabor filters on both layers with different scales and receptive field sizes.

Layer(s)	Accuracy
1	**21.4%**
2	13.0%

Table 6: Classification accuracy using different number of layers

4 Discussion

4.1 Local Contrast Normalization

As [6] pointed out, local contrast normalization increased the classification accuracy. Among two different methods subtractive normalization and divisive normalization, divisive normalization performed slightly better than subtractive normalization, and both outperformed the results using original pixel values. This might be because local normalization makes edges look more distinguished.

4.2 Filters

Although random filters are extremely easy to generate, they perform reasonably well. Moreover, in our experiments, randomized filters performed the best with softmax pooling. Theoretically, it is not clearly known what random filters are capable of. However, according to [6], random filters achieved almost 63% accuracy on Caltech-101 object recognition task, when used with non-linearities and proper pooling layers. This result implies that random filters operate in a rather complex way, but do produce surprisingly good results, and our results confirm this conjecture.

Gabor filters are known to be good at achieving translational invariance when combined with MAX-like pooling mechanism. Indeed, only with 48 Gabor filters, the network was able to correctly predict 21.4% of the test data.

Other type of filters we tried were the Gaussian and DoG filters. However, we found that even when combined with the MAX-like pooling mechanism, models with these filters did not perform as well as those with Gabor filters. Also, it is not clearly known what Gaussian filters can do when used in simple cells.

4.3 Pooling Mechanism

As [7] claimed, we could confirm that a MAX-like operation for the complex cells performed better than linear SUM operation. This result supports the idea that MAX-like operations are used globally in visual cortex, regardless of whether the data is a still image or a video.

4.4 Number of Layers

The network described in [8] had two layers of simple-complex cell-like sublayers. Furthermore, [6] found that architectures with two stages of feature extractions performed better on object classification that those with just one. Nonetheless, we weren't able to reproduce this observation in our experiments. One of the reasons may have been that we did not explore the parameter space sufficiently with the models we implemented.

4.5 Use of Temporal Information

We found that our temporal pooling extension actually decreased the classification results. We speculate that one of the reasons is because this pooling mechanism introduces another parameters, the number of frames to pool over, which we did not have very much time to tune. For future work, we feel that using temporal filters in the simple cell-like layer as in [2] would be more consistent with our pooling mechanism.

4.6 Limitations

Most importantly, our model does not involve learning, and relies only on the given structure and filters. Although we could take advantage of these customizable features of the network in order to investigate the nature of action recognition task in videos, we speculate that learning the filter banks ultimately should take place [6].

Secondly, our hard-wired filters were probably not tuned enough to produce the best result, or at least somewhat close to it. Because of the small

input size (100×100 pixels), the result of our simulation was very sensitive to the parameters such as the number of filters, the size, the area of receptive field, and so on.

Our experiments were also limited by the physical memory size of the machines; our machines didn't have large enough memory to hold big feature matrices, which in turn severely limited the size of the filter banks and the number of cells.

4.7 Possible Improvements and Future Works

We conjecture that the recognition accuracy can be boosted significantly by using the temporal relationships between adjacent frames. This is what we didn't exploit as much as possible in our experiments; our model produces the same results even if all frames are permuted in the same way. However, temporal context is very important since the same set of frames can be seen in two different actions depending on the order of the frames are shown. For example, a person sitting down can be viewed as a person standing up if we reverse the order of frames.

Also, as mentioned earlier, we think that learning the filter banks in either a supervised way or an unsupervised way can help the recognition task. To learn more complex features, it might be essential to stack more layers to the network ([7], [8]), although our experiments didn't show a strong evidence that this is the case.

There are much more ideas that we can try, and we will continue working on this project during this summer and try to achieve better results than the state-of-the-art result.

5 Acknowledgements

Thanks to Serena Yeung for providing us code base to build upon and to Quoc Le for technical help and advice on the project. Also, thanks to Tom Dean for leading CS 379C and providing us with a great insight on this field.

References

[1] P. Berkes, L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision* (2005), 5, 579–602.

[2] C. F. Cadieu, B. A. Olshausen. Learning Transformational Invariants from Natural Movies. *Advances in Neural Information Processing Systems (NIPS)*, 21:209–216, 2009. D. Koller and D. Schuurmans and Y. Bengio and L. Bottou, MIT Press, Cambridge, MA.

[3] A. Hyvarinen, J. Hurri, and P. O. Hoyer. Natural Image Statistics - A probabilistic approach to early computational vision. Springer-Verlag, 2009.

[4] M. Marszałek, I. Laptev, and C. Schmid. Actions in Context, *IEEE Conference on Computer Vision and Pattern Recognition*. 2009.

[5] H. Wang, M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. BMVC, 2009.

[6] K. Jarrett, K. Kavukcuoglu, and Y. Lecun. What is the Best Multi-Stage Architecture for Object Recognition?

[7] M. Riesenhuber, T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, November 1999.

[8] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.