Fixed Frame Temporal Pooling

John Thornton, Linda Main, and Andrew Srbic

Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia
{j.thornton,l.main}@griffith.edu.au andrew.srbic@griffithuni.edu.au

Abstract. Applications of unsupervised learning techniques to action recognition have proved highly competitive in comparison to supervised and hand-crafted approaches, despite not being designed to handle image processing problems. Many of these techniques are either based on biological models of cognition or have responses that correlate to those observed in biological systems. In this study we apply (for the first time) an adaptation of the latest hierarchical temporal memory (HTM) cortical learning algorithms (CLAs) to the problem of action recognition. These HTM algorithms are both unsupervised and represent one of the most complete high-level syntheses available of the current neuroscientific understanding of the functioning of neocortex.

Specifically, we extend the latest HTM work on augmented spatial pooling, to produce a fixed frame temporal pooler (FFTP). This pooler is evaluated on the well-known KTH action recognition data set and in comparison with the best performing unsupervised learning algorithm for bag-of-features classification in the area: independent subspace analysis (ISA). Our results show FFTP comes within 2% of ISA's performance and outperforms other comparable techniques on this data set. We take these results to be promising, given the preliminary nature of the research and that the FFTP algorithm is only a partial implementation of the proposed HTM architecture.

1 Introduction

Recent work on action recognition has shown that general purpose unsupervised learning techniques can outperform more specialised approaches that rely on hand-crafted feature detectors. In particular, Le *et al.* [9] used independent subspace analysis (ISA) to learn invariant spatio-temporal features for action recognition, and produced the best pre-2012 bag-of-features recognition rates on a range of well-known benchmark datasets. Subsequently, several more sophisticated hand-crafted and supervised learning approaches have outperformed ISA on two of these datasets (e.g. see [16]). Nevertheless, ISA remains the state-of-the-art within the class of unsupervised learning algorithms that do not rely on additional higher-level information concerning the form of the input.

ISA is an extension of independent components analysis (ICA), and, like ICA, is able to learn receptive fields similar to those found in the V1 and MT areas of visual cortex [6]. This makes ISA a significant benchmark for the evaluation of any pattern recognition system that attempts to model the functioning of visual cortex. One such system is the hierarchical temporal memory (HTM) architecture, originally proposed by Jeff Hawkins [4] and subsequently developed into a set of cortical learning algorithms (CLAs) [3].

In this paper we compare the performance of ISA with an adaptation of the latest version of the CLA spatial pooler. This pooler differs from ISA and ICA in being directly based on a biologically plausible learning model. The overall approach involves learning sparse distributed representations of the input by forming dendritic connections between functional units that model the minicolumns observed in human neocortex. These connections form in such a way that the column responses self-organise to produce sparse representations that have statistically interesting properties. A recent study has shown that an augmented CLA spatial pooler produces representations with significantly higher kurtosis values than those achieved by ICA on a well-known benchmark of greyscale images [15]. These higher values indicate a more evenly distributed response to the input data and a greater degree of statistical independence between the responses. Encouraging as such results are, they do not necessarily translate into better recognition rates. We therefore decided to compare this augmented spatial pooler with the state-of-the-art in the family of ICA algorithms: the ISA implementation of Le *et al.* [9].

To the best of our knowledge, no similar comparisons of the latest HTM cortical learning algorithms have been published. Instead, the majority of recent HTM papers still use the original NuPic implementation (e.g. [12]) made freely available by Numenta [2]. While some work has evaluated the question of parallelising the latest spatial and temporal pooler algorithms [11], there is no work we know of that tests these algorithms on standard vision processing tasks, such as action recognition.

The research here extends the previous statistical evaluation of the spatial pooler in [15], by adapting it to represent sequences of frames from movies. This is achieved by means of additively superimposing the responses to individual frames. Following the evaluation methodology used by Le *et al.*, we limited this superimposition to sequences of ten frames, producing a fixed frame temporal pooling (FFTP) algorithm that is considerably simpler than the temporal pooler proposed in [3]. We then compared the performance of FFTP to ISA on the well-known KTH action recognition data set. Our results show that FFTP comes within 2% of ISA's performance and outperforms the other directly comparable algorithms. We take this as a promising result given this is the first application of an HTM spatial pooler to the domain of moving images, and also given that the spatial pooler is only a single component within the HTM architecture. This means that the study is only a preliminary indication of the potential performance of a complete HTM implementation.

In the remainder of the paper we provide background on the original HTM spatial pooler and describe the enhancements proposed in [15]. We then introduce the fixed frame temporal pooling algorithm and evaluate its performance compared to ISA on the KTH dataset. These experiments follow the protocols developed by Wang *et al.* [17] that were used in the original ISA study [9].

2 Spatial Pooling

2.1 The HTM Architecture

The HTM model was originally introduced by Jeff Hawkins [4, 5] and provides a framework within which the functioning of mammalian neocortex is understood in terms of a single algorithmic process. That process is a form of hierarchically-structured Bayesian-like predictive inference. The hierarchy itself consists of a set of regions, where each region consists of a set of columns. These columns correspond to the mini-columns observed within neocortex that Mountcastle identified as the basic functional units of neocortical processing [10]. Each column in turn consists of a set of neurons and their associated dendrites and synapses. According to HTM theory, these neurons control which columns in a region are currently active, and which columns are currently predicting they will become active.

Within each region, two functions are combined: a spatial pooler to form sparse representations of input, and a temporal pooler to learn temporal sequences. The two pooling functions form a single processing unit which is then replicated and arranged in a hierarchical structure. In this way, the HTM architecture is able to learn and exchange inferences about temporal sequences rather than just spatial patterns.

2.2 HTM Spatial Pooling

From the perspective of spatial pooling, a column can be considered as a unified entity with an associated set of proximal dendrites that synapse directly with the input (see [3]). These synapses are not associated with weights that multiplicatively determine the strength of the signal. Instead, each dendrite is associated with a potential synapse and each synapse is associated with a permanence value. If the permanence value of a synapse passes a certain threshold then the synapse is connected and the dendrite will directly relay the input to which it is connected, otherwise the synapse remains potential and inactive. The column then sums the inputs from all its connected synapses to determine its level of activity.

A spatial pooler learns on the basis of how well the synapses from a particular column match (or overlap) the input to which the synapses are connected. Instead of altering the relative weights of the synapses of neighbouring columns, a strongly activated column will compete with and inhibit its less active neighbours, implementing the function of short-range inter-columnar inhibition neurons [14]. At the end of this process, only the potential synapses belonging to the winning columns that best represent the current input will be able to learn. Here learning entails increasing the permanence values of potential synapses that are connected to active input and decreasing the permanence values of those connected to inactive input. This implements the forming and un-forming of synaptic connections discussed above.

2.3 Augmented Spatial Pooling

The augmented spatial pooler (ASP) introduced in [15] alters the learning algorithm of the original HTM spatial pooler in order that stable representations of greyscale images can be formed. The original spatial pooler was designed to encode binary input and failed to reliably converge on greyscale input. This original learning algorithm additively boosts the response of columns whose average *activity* falls below a predefined threshold (otherwise the boost value is set to one) and increases the permanence values of all potential synapses of columns whose *response* falls below the same threshold. The augmented spatial pooler will similarly boost the output of all columns whose activity falls below the threshold but only up to a predefined maximum boost value. Once that value is reached, the column boost is reset to one, and the closest disconnected column synapse with the largest permanence value has that value increased to the point where it becomes connected. This forcing of a connection significantly improves the convergence behaviour of the pooler on both binary and greyscale input.

Figure 1 shows a matrix of 16×16 ASP columns and their associated synapse connections after training on a set of grayscale images (see [15] for details). Here each of the 256 squares represents a column, and each of the 256 pixels in an individual square represents a synapse belonging to the associated column. A disconnected synapse is represented by a black pixel, otherwise all non-black pixels represent connected synapses, where the intensity of a pixel represents the boost value of the associated column. This diagram illustrates the relative sparsity and simplicity of the image filters produced by a set of columns and should be contrasted with the correspondingly more complex filters produced by ISA (for example see Figure 2 from [9]).

3 Fixed Frame Temporal Pooling

The temporal pooler proposed in the most recent HTM technical report [3] implements a new column structure consisting of four cells, where each cell is laterally connected to other cells within the same region. These connections again consist of synapses whose permanence values are learnt according to the responses of the spatial pooler and the subsequent activity of the temporal pooler. The task of the temporal pooler is to predict the sequence of columns that will become active in the immediate future, according to the current and past activity of the columns. This prediction produces a pattern of responses where the currently active columns are co-active with the columns that are expected to become active in succeeding time intervals (see [3] for details).

In the current study we are interested in a direct comparison between the sparse representations produced by ASP and the responses of the ISA filters. Originally ISA was developed to handle static images, and was converted to handle moving images by altering the *form* of the input. This involved taking static image patches (typically 16×16 pixels) and arranging them in *temporal stacks.* These stacks consist of ten patches that represent a time ‘tunnel’ through a video, where the first patch is taken from location x_1, y_1 of the first video frame,

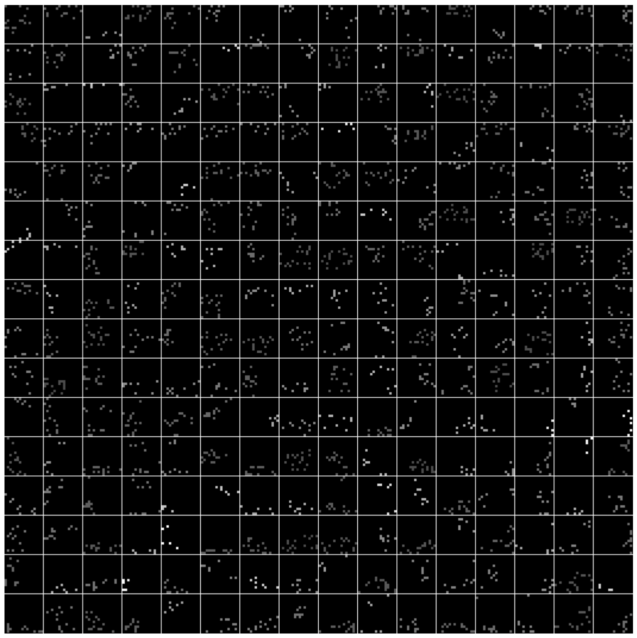

Fig. 1. Image filters for 256 ASP columns after training on a set of greyscale photographs of natural scenes (taken from [15]). Each square represents a column and each dot within a square represents a connected synapse.

the second patch is taken from location x_1, y_1 of the second video frame, and so on. These ten patches are then 'flattened' to form a single image that becomes the input to an ISA unit. The units themselves are convolved to overlap the entire video frame (see [9] for full details).

This means that ISA does not predict a particular action as the action unfolds. Instead it computes its responses retrospectively. In order to compute a comparable representation using ASP, rather than flattening a temporal input stack to form a single image, we decided to additively superimpose the ASP responses to each image patch in a ten patch stack, producing a single response for the entire stack (see the bottom right of Figure 2 below). This form of temporal pooling thereby combines the ASP responses to a *fixed* sequence of temporally ordered video frame patches (hence the name *Fixed Frame Temporal Pooling* or FFTP).

The FFTP algorithm is of a different and lower order of complexity compared to the full temporal pooler, in that FFTP performs no further learning, and simply combines the outputs of the spatial pooler in the temporal dimension. However, the representations of the two temporal poolers will still be similar. The main differences are that the full temporal pooler will *dynamically* predict a future sequence of frames and that the number of frames encompassed in that prediction will *vary* according to the strength of the learned connections. In contrast, FFTP firstly has the future data already available and so is not required to predict future input, and secondly, it has the time dimension of its representation fixed in advance (in this case to ten consecutive frames).

However, in combining the responses of the spatial pooler to form a temporal representation there are several possibilities in terms of how the column outputs are measured. Firstly, consider a spatial pooler comprising 16×16 columns arranged in a two-dimensional grid that corresponds to a 16×16 pixel input patch. Once the spatial pooler has converged it will produce stable responses to given input patches (i.e. the same set of columns will become active given the same input pattern). This response forms a sparse distributed representation of the original patch that then becomes the input for the temporal pooler. Given ten input patches we obtain ten corresponding sparse representations that are then combined to form a single 16×16 temporally pooled response (as per the diagram on the bottom right of Figure 2). There are now three possible ways we can combine these column activities:

1. Each column is represented using a binary code, where 0 indicates the column has been inactive during the entire ten frame sequence, and 1 indicates the column has been active at least once. We term this the *Binary* code.
2. Each column is represented according to a $0 - 10$ count of how many times it has been active during the entire ten frame sequence. We term this the *Count* code.
3. Each column is represented according to the *Count* code *multiplied* by the boost value for that column. We term this the *Boost* code.[1]

The pseudocode detailing the calculation of these three measures is shown in Algorithm 1. Here *code* specifies the measure and *pooledOutput* specifies the temporally pooled column outputs that form the input to Wang *et al.*'s pipeline (described in Section 4).

Figure 2 gives a pictorial illustration of the functioning of FFTP. The first ten images on the bottom row show the augmented spatial pooler's response to the static image patch appearing immediately above each response. For example, the bottom left response represents the outputs of the 256 columns, arranged in a 16×16 matrix. If a column is inactive it is represented as a black pixel, otherwise the boost activity is represented by the lightness of the pixel intensity. Note that these images differ from those in Figure 1, in that now each pixel represents whether or not a *column* is *active,* whereas in Figure 1 each pixel represented whether or not a *synapse* for a particular column is *connected.*

The ten images on the second row up from the bottom of the figure form a stack of temporally consecutive instants taken from a KTH movie of a man jogging. A complete frame from this movie is shown in the top right of the figure.

[1] The boost value for the column represents the amount the output of the column needs to be increased in order for it to exceed the activity threshold described earlier. It is this boost value in combination with the growth of new synapses that ensures the spatial pooler representations are distributed relatively evenly across all columns.

Algorithm 1 FixedFrameTemporalPooler(*columns*, *stacks*, *code*, *boost*)

```
for each $c$ in $columns$ do $pooledOutput(c) = 0$
for each $s$ in $stacks$ do
  for each $patch$ in $s$ do
    for each $c$ in $columns$ do
      if $c$ is active in $patch$ then
        if $code$ is $Binary$ then $pooledOutput(c) = 1$
        else $pooledOutput(c) = pooledOutput(c) + 1$
      end if
    end for
  end for
  if $code$ is $Boost$ then
    for each $c$ in $columns$ do $pooledOutput(c) = pooledOutput(c) \times boost(c)$
  end if
end for
return $pooledOutput$
```

The ten images themselves show the man jogging through the patch appearing immediately to his left in the complete frame. The overall output of the FFTP algorithm for this image stack is represented in the eleventh image at the bottom right of the figure. This shows the summed value of the ten responses (using the *Boost* code). It is these summed values that become the input to Wang *et al.*'s pipeline in the experimental study.

The graph in the top left of Figure 2 shows how the number of active columns varies in proportion to the amount of activity that is occurring in the image. Here each point in the graph represents a count of the number of columns that respond to a particular stack of image patches. The stacks are ordered consecutively in time and represent what occurs in each of the three patches shown in the complete frame, i.e. as the man jogs across the field. Note the borders of the patches on the complete frame are the key to the graph plots. For example, the patch appearing at the bottom of the complete frame is represented by the dotted line in the graph that has the lowest column counts. These low counts show that very little change occurs in the corresponding patch as the man jogs across the field. In contrast, the peaks on the other two plots correspond to those portions of the movie where the man jogs through the patches that the plots are representing. More specifically, the plot shows the man jogging across the field four times, first from right to left, then left to right, and so on. The first graph peak therefore corresponds to the man first passing through the right patch, the second to his passing through the left patch, the third to his running back and passing through the left patch, then the right, and so on. This shows how the number of active columns is correlated to the number and degree of change occurring in the image pixels.

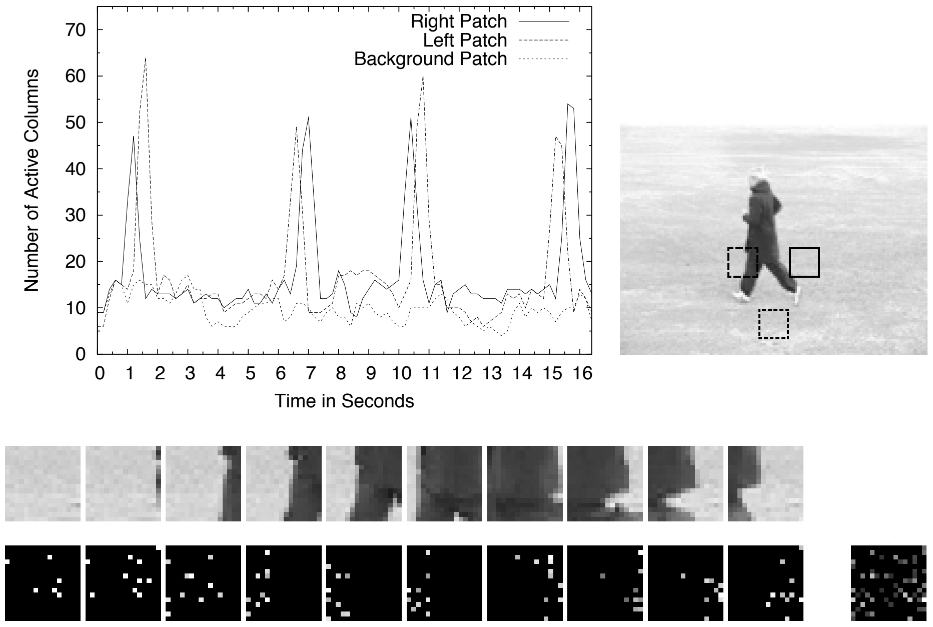

Fig. 2. Top left: Graph showing the number of active spatial pooler columns per unit time as the pooler processes a movie of a man jogging back and forth across a field. **Top right**: A full frame from the movie, showing the three patches from which the graph data is collected. **Second row from bottom**: a sequence of ten image patches taken from the left-hand patch position of the full frame image. **Bottom row**: Column responses of the FFTP algorithm where each response corresponds to the image immediately above the response. **Bottom right**: The superimposed stacked code produced from the ten preceding responses (this is the final output of the FFTP algorithm).

4 Experimental Study

In order to make direct comparisons between our FFTP encodings and previous work in the area, we decided to use Wang *et al.*'s pipeline [17][2] (the same pipeline is reported in the Le *et al.* ISA study). This pipeline takes the features determined by an external feature detection algorithm (e.g. ISA or FFTP), performs K-means vector quantisation, and finally classifies using a χ^2 kernel SVM for each action category.

We further decided to evaluate our algorithm on the widely studied KTH dataset [13].[3] This set contains greyscale videos of six human action types (walking, jogging, running, boxing, handwaving and handclapping). The actions are performed several times by 25 different people, and recorded in four different

[2] Available at http://au.stanford.edu/∼wzou/
[3] Available at http://www.nada.kth.se/cvap/actions/

settings; three outdoors and one indoors, making an approximate total of 600 videos, each containing an average of 484 frames. The videos are divided into a test set: (persons 2, 3, 5, 6, 7, 8, 9, 10, and 12) and a training set (the remaining 16 people).[4] We chose this dataset because it is one of the most widely reported in the literature and, as it is encoded in greyscale, it did not require any modifications to the existing augmented spatial pooler (see Figure 2 for an example KTH movie image).

Following the protocol used in [9] and [17], we trained the augmented spatial pooler on 200,000 randomly selected stacks of ten 16×16 pixel patches, densely sampled with a 50% overlap horizontally, vertically, and temporally (note: sampling was set to avoid selecting sequences that went over the end of a movie clip). The trained spatial pooler was then used to generate stacked responses to the full set of image patches, and these stacked codes became the input for Wang *at al.*'s pipeline. Within the pipeline, the K-means clustering algorithm was trained on 3,000,000 of the ASP stacked codes (generated from the original training split) to form 3,000 centroids (identical settings were used to generate the ISA results). Mean accuracy was then calculated by averaging the accuracy of six SVM classifiers (one for each action class), over three runs of K-means, using stacked codes generated on the original test set (again following the protocol used in [9] and [17]).

Table 1 presents the accuracy results for the three versions of FFTP tested, for ISA, and for the four other comparable techniques reported in Wang *et al.*'s study [17]. These results firstly show that FFTP (using *Boost*) comes close to matching ISA's performance (falling short by 1.7%). To verify this difference we performed a one-tailed independent two-sample t-test on the three accuracy results for FFTP *Boost* and for the three ISA results.[5] This confirmed the hypothesis that the mean accuracy of ISA is greater than the mean accuracy of FFTP *Boost* at a 1.5% level of significance.

It should be noted that in order to make a direct comparison between FFTP and ISA we did not test the second level hierarchy developed by Le *et al.* or use their norm-thresholding heuristic (see [9] for details). Nevertheless, we obtained slightly better results for ISA (94.5% versus 93.9%) in comparison to those reported in [9] (where both the hierarchy and the thresholding were employed).

Table 1 also shows that, of the three FFTP coding methods tested, *Boost* has the slightly better performance, followed by *Count*, with *Binary* coming third. This indicates that more fine grained information concerning the level of activity in each column provides useful information to the classifiers and contributes to better recognition rates. However, given the small sample sizes on which the average accuracy is based, these relative differences did not turn out to be statistically significant.

Lastly, FFTP shows notably improved performance over the four other comparable techniques that use dense sampling, reported in the Wang *et al.* study (HOG3D [7], HOG/HOF, HOG and HOF [8]). However, if we widen our con-

[4] This training/test set split follows the protocol defined in the Appendix of [9].
[5] FFTP *Boost* accuracy standard deviation was 0.3086 and ISA was 0.1782.

Table 1. Average accuracy on the KTH data set. The FFTP and ISA results were produced by us using Wang *et al.*'s pipeline. The other results (HOG3D [7], HOF and HOG [8]) were the comparable algorithms with dense sampling produced by Wang *et al.* in [17].

	FFTP			Dense				
Algorithm	Boost	Count	Binary	HOG3D	HOG/HOF	HOG	HOF	ISA
Accuracy %	**92.8**	92.7	92.4	85.3	86.1	79.0	88.0	**94.5**

sideration to include supervised learning techniques and techniques that employ interest point detectors, there have been some notable developments since the publication of Le *et al.*'s 2011 study. For example, Wang *et al.* [16] have developed a new supervised dictionary learning technique that outperforms ISA by 0.27%, meaning ISA is no longer the state-of-the-art on KTH. Nevertheless, it remains the best *unsupervised* learning technique that does not use interest point detectors, and so remains the most relevant technique for an evaluation of FFTP. This is because FFTP is also a pure unsupervised learning technique that plays the role of a *component* within an HTM architecture. The HTM architecture itself is designed to learn higher level features (without supervision) and then to feedback this higher level contextual information to the lower level spatial and temporal pooler representations. Following the principle of comparing like with like, it would not be equitable to compare an HTM component with Wang *et al.*'s new supervised learning architecture. Instead, it is the aim of this study to evaluate FFTP as a way of encoding image patches for *subsequent* recognition (hence the use of Wang *et al.*'s pipeline), rather than evaluate the recognition performance of an entire HTM. That task we leave for future work.

5 Discussion and Conclusions

The fact that FFTP produces results that are closely comparable with ISA is encouraging. Firstly, we should consider that this is the first application and evaluation of the new HTM spatial pooler on action recognition in the literature. FFTP is also unusual in not using a mathematical technique to optimise an objective function. It rather distributes the codes across columns according to a biologically inspired procedure of lateral inhibition and competition. This makes the system more adaptable than an optimiser, as it can adjust its representations in real time according to the input it receives (this kind of flexibility is also observed in the neurodynamics of natural systems [1]). In addition, unlike standard neural network algorithms, FFTP learns by forming and un-forming synaptic connections instead of adjusting the weights of those connections. This procedure receives support from the the observation that real synaptic transmission is too stochastic to provide the kind of fine distinction that many artificial neural network algorithms require [14].

It should also be remembered that the ISA algorithm is the result of nearly two decades of intensive research into the development of ICA related techniques. In contrast, the HTM spatial pooler was proposed in 2010 and is only now being applied to action recognition. We therefore have a reasonable expectation that the performance of the family of HTM poolers will improve over time.

Next, we should emphasise that FFTP is a partial implementation of the full temporal pooler proposed in [3]. As such it only combines the output of the spatial pooler according to an artificial boundary of ten frames. The full temporal pooler is designed to learn common sequences of actions of variable length, and thereby discover for itself when a particular sequence starts and stops (according to how often such sequences are repeated as input). This more natural encoding should be more efficient and more representative of the underlying structure of action events. If such encodings can be produced, it is again reasonable to expect a positive effect on event recognition rates.

In conclusion, the current study represents a preliminary evaluation of a promising new technology. In that context, our having nearly matched the performance of the state-of-the-art in general purpose unsupervised learning is a strong indication that it is worth developing the technology further. Clearly, the next step is to develop and test a full (*non*-fixed frame) temporal pooler on the action recognition data sets and, if that proves successful, to build a hierarchy of such poolers.

References

1. Freeman, W.J.: How brains make up their minds. Columbia University Press, New York (2000)
2. George, D., Jaros, B.: The HTM learning algorithms. Tech. rep., Numenta, Inc, Palto Alto (2007), www.numenta.com/htm-overview/education/Numenta_HTM_Learning_Algos.pdf
3. Hawkins, J., Ahmad, S., Dubinsky, D.: Hierarchical temporal memory including HTM cortical learning algorithms. Tech. rep., Numenta, Inc, Palto Alto (2010), www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf
4. Hawkins, J., Blakeslee, S.: On intelligence. Henry Holt, New York (2004)
5. Hawkins, J., George, D.: Hierarchical temporal memory: Concepts, theory and terminology. Tech. rep., Numenta, Inc, Palto Alto (2006), www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf
6. Hyvärinen, A., Hurri, J., Hoyer, P.: Natural Image Statistics: A probabilistic approach to early computational vision. Springer-Verlag New York Inc (2009)
7. Kläser, A., Marszalek, M., Schmid, C.: A spatio-temporal descriptor based on 3D-gradients. In: British Machine Vision Conference, BMVC 2008. pp. 995–1004 (2008)
8. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08). pp. 1–8 (2008)
9. Le, Q., Zou, W., Yeung, S., Ng, A.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: Proceedings

of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11). pp. 3361–3368 (2011)

10. Mountcastle, V.B.: Introduction to the special issue on computation in cortical columns. Cerebral Cortex 13(1), 2–4 (2003)
11. Price, R.W.: Hierarchical Temporal Memory Cortical Learning Algorithm for Pattern Recognition on Multi-core Architectures. Master's thesis, Portland State University (2011)
12. Rozado, D., Rodriguez, F.B., Varona, P.: Extending the bioinspired hierarchical temporal memory paradigm for sign language recognition. Neurocomputing 79, 75–86 (2012)
13. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: Proceedings of International Conference on Pattern Recognition (ICPR 2004). pp. 3361, 3362, 3366 (2004)
14. Stuart, G., Spruston, N., Häusser, M.: Dendrites. Oxford University Press, New York (2008)
15. Thornton, J., Srbic, A.: Spatial pooling for greyscale images. International Journal of Machine Learning and Cybernetics 2, 1–10 (2011)
16. Wang, H., Yuan, C., Hu, W., Sun, C.: Supervised class-specific dictionary learning for sparse modeling in action recognition. International Journal of Machine Learning and Cybernetics 2, 1–10 (2012)
17. Wang, H., Ulla, M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: British Machine Vision Conference, BMVC 2009. pp. 127–138 (2009)