Cortically-Inspired Overcomplete Feature Learning for Colour Images

Benjamin Cowley, Adam Kneller, and John Thornton

Institute for Integrated and Intelligent Systems,
Griffith University, QLD, Australia
{benjamin.cowley,adam.kneller}@griffithuni.edu.au
j.thornton@griffith.edu.au

Abstract. The Hierarchical Temporal Memory (HTM) framework is a deep learning system inspired by the functioning of the human neocortex. In this paper we investigate the feasibility of this framework by evaluating the performance of one component, the spatial pooler. Using a recently developed implementation, the augmented spatial pooler (ASP), as a single layer feature detector, we test its performance using a standard image classification pipeline. The main contributions of the paper are the implementation and evaluation of modifications to ASP that enable it to form overcomplete representations of the input and to form connections with multiple data channels. Our results show that these modifications significantly improve the utility of ASP, making its performance competitive with more traditional feature detectors such as sparse restricted Boltzmann machines and sparse auto-encoders.

Keywords: Hierarchical Temporal Memory, Biologically-Inspired AI, Image Classification, Machine Learning

1 Introduction

One of the most significant developments in contemporary computational neuroscience is the emergence of a unified understanding that the basic function of the neocortex is to form predictions concerning the structure of its own future input. This approach has resulted in Friston's proposal that free energy minimisation is the fundamental organising principle behind all cognition [6] and that such minimisation is best achieved within a hierarchical predictive coding (HPC) architecture [2]. In this paper we examine a closely related model of neocortical processing that fits within the basic paradigm of free energy minimisation but differs from HPC in forming predictions by means of hierarchical sequence learning rather than by means of feedback with predictive coding error units. This Hierarchical Temporal Memory (HTM) model of neocortical processing was first proposed by Jeff Hawkins in his 2004 book *On Intelligence* [8] and has subsequently been partially implemented in a series of cortical learning algorithms proposed by Hawkins and his colleagues [9, 7], and developed further in related research [22, 23].

The focus of this paper is the investigation and further improvement of one component of the HTM model, the spatial pooler (SP). Specifically, we investigate, for the first time, the performance of the spatial pooler as a feature detector on colour images. In a complete HTM implementation, the spatial pooler is one of two basic components, the other being the temporal pooler (TP). The primary function of the SP is to form sparse distributed representations of the input data that are then combined into temporal sequences by the TP and passed up to the next level of the HTM hierarchy. At this next level an identical SP sparsifies the input from the lower level and passes this to a second identical TP. This entire process is then repeated at each level of the hierarchy until the apex is reached. The techniques employed by the SP and TP algorithms exploit many key neuroscientific ideas, including the implementation of mini-columns [18], inhibition [17], probabilistic connections to inputs [21] and geometric distances governing the relationship between nodes and connections [12]. However, in a fully functional HTM there would also be feedback between levels, such that higher level sequence information is able to modify lower level sequences via a process of Bayesian-like belief revision until an equilibrium state is reached that represents the most probable causes of a given input stream and that predicts the most probable trajectory of that stream. *How* such feedback will work is still a matter for further research.

The augmented spatial pooler algorithm (ASP) [23] used in this study extends the original SP proposed by Hawkins *et al.* [7] in having the ability to encode non-binary data. ASP was also shown to have superior convergence behaviour in comparison to the original SP, with the generated sparse codes exhibiting similar properties of statistical independence to those found in biological neural codes [23, 15]. In related work, ASP has shown promise as a single layer unsupervised feature detector for use in classification of greyscale videos [22] and handwritten digits [15]. In this paper we further evaluate the utility of ASP as a single layer unsupervised feature detector by applying it to the domain of natural colour image classification. Our main innovations are to extend the algorithm by allowing it to form overcomplete representations of the input data and by allowing individual coding units to form connections with multiple levels of data input. These connections are then used to learn activation patterns across multiple colour channels whilst still maintaining the topographic relationships inherent in the original colour images.

In order to evaluate these innovations, we tested ASP on the CIFAR-10 dataset [14] adopting the same encoding and classification methodology used by Coates *et al.* in 2011 [3]. This methodology allowed us to easily analyse how modifications to the encoding/classification framework affect the relative performance of ASP. We also experimented with changes to the patch size, varying the number of detected features, whitening the input data, and altering the ASP hyper-parameters governing the sparsity of the output and the probability of connections to input elements. Overall, the experimental study showed that our modifications significantly improved both the performance and the utility of the original ASP as a feature detector for colour images, making it competitive with a range of more traditional approaches, such as the restricted Boltzmann machine and the sparse auto-encoder.

In the remainder of the paper we provide a lower level description of the ASP algorithm and the modifications that form the main contribution of this research. We then provide details of the experimental design, present our results and discuss their significance.

2 HTM Augmented Spatial Pooler

Within the Hierarchical Temporal Memory (HTM) model, the primary functional unit is the *column*. This component is based on the mini-column found in the neocortex [18]. Each HTM column contains a collection of coding units, the majority of which form part of the temporal pooler (TP) and function as sequence learners. A column's spatial pooler component consists of a single coding unit that is activated entirely through connections made with bottom up input and acts as a first order feature detector. As each column contains a single SP coding unit and as we are not including temporal pooler coding units in the current study, from now on we will refer to SP coding units as *columns*.

Unlike many other unsupervised feature detectors, such as restricted Boltzmann machines, auto-encoders and independent components analysis, the geometric relationships between input data elements and ASP columns play a key role in the learning and encoding processes. The columns are laid out in a matrix and receive input which is also formatted in a matrix. These matrices share the same topology, such that a column located at entry $(2, 4)$ of the column matrix will have a distance of zero to an input element located at entry $(2, 4)$ of the input matrix.

Columns connect to the input through binary (i.e. unweighted) *synapses* that are either connected or unconnected. As with connections between layers in the neocortex [12], columns are more likely to be connected to input elements which have a smaller distance to the column [15]. In addition to the the use of geometric distances, another way that ASP differs from many traditional feature detectors is that the columns are not fully connected to the input. This again reflects the situation in the neocortex, where cortical neurons are only partially connected to their associated receptive fields [21]. In ASP this partial connectivity is implemented during initialisation, where input elements with which a column can potentially connect are chosen at random with a probability p. During our experiments we test the effect of changing p, as reported in Section 4.2.

The sparsity of the ASP output codes is produced through a process of inhibition. This process has conceptual similarities to the function of inhibitory cortical neurons, such as basket cells, which can synapse directly to the soma of neighbouring neurons and whose action potentials cause connected neurons to be less likely to produce action potentials [13]. In ASP, inhibition is a competitive process in which columns with a higher level of activity reduce neighbouring column activities to zero. The inhibitory range of a column is determined by the average synapse receptive field size over all columns, which in turn is determined by the area covered by a column's connected synapses. Within this inhibitory range, a column is fully connected to all of its neighbouring columns. This has a biologically plausible grounding in that inhibitory neurons have extremely high local connection densities [5]. The number of columns that a column can inhibit is inversely proportional to the *desired level of activity*, which represents the preferred fraction of columns active during encoding.

During training, ASP learns features through the modification of the *permanence* values of a column's synapses. If the value of an input element with which a column synapses is greater than the average across the input matrix then the permanence value is increased, otherwise it is decreased. Moreover, if the permanence value falls below a predefined threshold, then the synapse becomes disconnected. This learning strategy is directly inspired by Hebbian learning, where neurons are more likely to form synapses with other neurons which are active at the same time [10]. To aid learning a *boost* multiplier is applied to columns with a low activity level. Finally, if the boost fails to activate a column sufficiently, then the column will grow a new synapse by increasing the permanence value of a previously disconnected potential synapse above the threshold (for a more detailed explanation of the ASP algorithm see [23]).

2.1 Multiple Input Matrices

To allow ASP to learn and encode data from multiple sources we have implemented a multiple input paradigm where a column can form synapses to multiple input matrices. These matrices share the same topographic relational properties to the column matrix, such that in a system with two input matrices, an input element at entry (2,4) of one input matrix would have the same distance to a random column as an element in entry (2,4) of the second input matrix. This is illustrated in Figure 1.

In our experiments we use this input paradigm to split the colour data across three input matrices, with one input matrix for each of the RGB channels. In this way columns can learn features incorporating the three colour channels while preserving the pixels' topographic relationship vis-à-vis the column matrix and other pixels. We compare the performance of this method to that of a method which does not maintain these topographic relationships in Section 4.1.

2.2 Overcompleteness

In previous implementations of ASP, the number of input columns was limited to be less than or equal to the number of inputs. Although ASP performed well under these constraints [22, 15] by allowing the number of columns to exceed the number of inputs, ASP can learn an overcomplete feature set of the input data. Based on empirical observation of the V1 area of visual cortex [19], we conjecture that increasing the feature set size may allow for better representations of the variances in complex data, such as natural images.

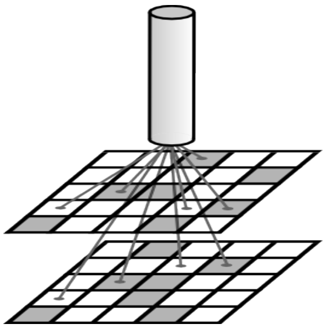

Fig. 1. Diagram of a column connecting to two input matrices.

To properly implement overcompleteness in the ASP algorithm we must also maintain the topographic relationships between the columns and the input matrices. To solve this problem we stack the columns on top of each other, with multiple columns sharing the same topographic location. Initialisation of the columns' location is accomplished by applying Algorithm 1, which is sufficient for a square-shaped matrix of any size. In this algorithm the first column is placed in the centre of the column matrix and subsequent columns are arranged by spiralling away from the centre until the column matrix is filled. We recursively call the function until all columns have been placed.

3 Feature Extraction and Classification Framework

For our experiments we decided to adopt the feature learning, extraction and classification framework first developed by Coates *et al.* [3]. In this work, the authors evaluated a number of unsupervised feature learning algorithms including sparse Restricted Boltzmann Machines (RBMs), sparse auto-encoders, k-means and k-means clustering with 'triangle' activation. These algorithms were tested on patches taken from a collection of image datasets and were analysed on the basis of varying the following settings: (i) the number of features used in the algorithm, (ii) the 'stride' used in obtaining patches from the images (the number of pixels between each patch), (iii) the 'receptive field size' (the size of each patch taken from the images), and (iv) the effect of preprocessing the data by whitening the images. After feature extraction using the learning algorithms the data was pooled to reduce dimensionality and classified using a linear SVM.

Following Coates *et al.* allows us to test the ASP within a simple framework designed for unsupervised single layer feature detectors. Additionally we can easily modify settings, such as the feature number and patch size, to investigate how these changes affect ASP and compare the results to other algorithms within the context of the same framework. In the remainder of this section we introduce the key steps of this framework.

Algorithm 1 initialiseColumnLocation($columnQueue, width, height$)

```
 1: $x \leftarrow \lceil width \div 2 \rceil$
 2: $y \leftarrow \lceil height \div 2 \rceil$
 3: $setColumnLocation(pop(columnQueue), x, y)$
 4: $count \leftarrow 1$
 5: $xInc \leftarrow 1$
 6: $yInc \leftarrow 0$
 7: loop
 8:    for $i \leftarrow 0$ to $count$ do
 9:       $x \leftarrow x + xInc$
10:       $y \leftarrow y + yInc$
11:       if $isEmpty(columnQueue)$ then
12:          return
13:       end if
14:       if $x < 0$ or $x \geq width$ or $y < 0$ or $y \geq height$ then
15:          $initialiseColumnLocation(columnQueue, width, height)$
16:          return
17:       end if
18:       $setColumnLocation(pop(columnQueue), x, y)$
19:    end for
20:    if $yInc = 0$ then
21:       $yInc \leftarrow xInc$
22:       $xInc \leftarrow 0$
23:    else if $xInc = 0$ then
24:       $xInc \leftarrow -yInc$
25:       $yInc \leftarrow 0$
26:       $count \leftarrow count + 1$
27:    end if
28: end loop
```

3.1 Feature Learning and Extraction

The first step is to train the unsupervised feature learning algorithms. This training is performed on 400,000 randomly sampled patches from the training set (8 from each image, taking all three colour channels). Using the trained algorithms we then extract features from each image in a convolutional manner, one patch at a time by 'sliding' across and down the image by a stride value. On the basis of the the clear results of [3] in our experiments we only use a stride of one pixel. Before feature learning and extraction the patches may be preprocessed with a whitening algorithm.

3.2 Dimension Reduction and Classification

Before classification, the feature activations are pooled to reduce dimensionality. This is achieved by a simple sum-pooling algorithm where the image is divided into quadrants (top-left, top-right, bottom-right and bottom-left). Each of the quadrants is represented by a vector of size f, where f is the number of features extracted by the feature learning algorithm. The activities for each individual feature are summed across each patch within its corresponding quadrant. The summed value for each feature is stored within the quadrant vectors which are then concatenated. Thus each image is represented by a $4f$ sized real-valued vector. We then perform classification of the images using a simple 1-vs-all linear SVM.

4 Experiments and Results

We performed a number of experiments to analyse the performance of ASP on colour natural images. For all experiments we use the CIFAR-10 dataset of colour natural images [14]. CIFAR-10 is a popular dataset for computer vision benchmarking comprising 60,000 32×32 pixel RBG images. The dataset contains 10 separate classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck, pre-divided into a training set of 50,000 images and a testing set of 10,000 images.

When experimenting with different parameter settings for ASP and different configurations of the feature extraction and classification framework we ran multiple tests to ensure that the results we obtained are representative. To this end, we used two different randomly sampled sets of patches to train ASP, and two different initialisation seeds for ASP for each of the sets; making a total of four tests for each experimental setting (except where noted). When classifying each of these four tests with the SVM we used the same C hyper-parameter that was found to be the best through optimisation.

4.1 Handling Colour

We first present the results of experiments designed to evaluate the effect of maintaining topological relationships of the input pixel colour channels using our multiple input matrix method, versus only partially maintaining these relationships.

The number of ASP columns used in this round of experiments is 108. This is equal to number of input elements (6×6 pixels across 3 colour channels). When using multiple input matrices we use three input 6×6 matrices, one for each colour channel. For the single input matrix we attempt to preserve the topological relationships by grouping the three channels together and interlocking adjacent pixel groups, as shown in Figure 2. This process results in a 9×12 input matrix.

For both methods we ran a total of 20 tests, using the two sample sets and ten initialisation seed values. Across the 20 tests the single matrix input configuration produced an average classification accuracy of 59.49% while the multiple matrix input configuration produced an accuracy 60.58%. A one-tailed t-test ($p < 0.01$) verified that the multiple matrix input method's higher average classification accuracy is statistically significant.

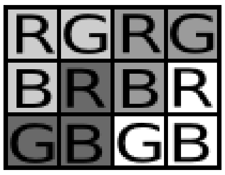

Fig. 2. Diagram of RBG pixel input data using a single matrix. Groups of three squares with the same level of shading represent the R, G and B channels of the same underlying pixel. The diagram therefore encodes the RGB channels of four (2×2) underlying pixels in a 4×3 matrix.

4.2 Probability of Input Connections

To evaluate the utility of a column having a sparse connection to the input matrices, we experimented with adjustments to the *p* hyper-parameter. This parameter governs the probability that a connection will be formed with any given input element, as discussed in Section 2.

In these experiments we used the multiple input matrices method, using a patch size of 6×6 and 200 columns. We ran tests using *p* values ranging from 0.05 to 0.8 in steps of 0.05. Given that the higher classification accuracies were between 0.1 and 0.2 we ran further tests between 0.1 and 0.2 in steps of 0.01. The results from these tests are graphed in Figure 3.

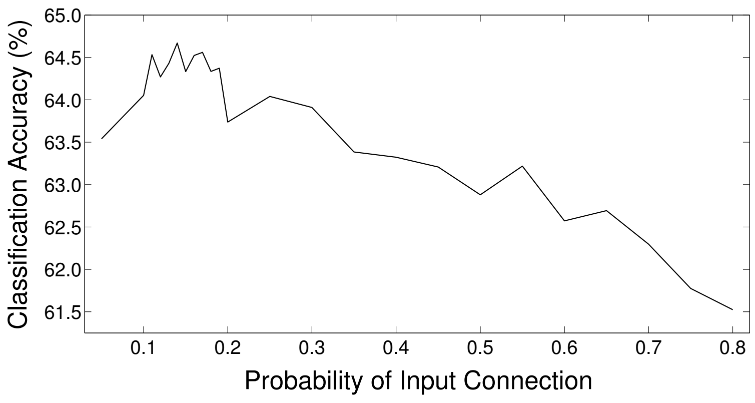

Fig. 3. Average classification accuracy with changes to the probability of input connection or *p* hyper-parameter value.

As can be seen from Figure 3, the accuracy has a broad 'peak' between 0.1 and 0.2, before trending lower as the value of *p* increases. The highest accuracy was achieved using a value of 0.14 with an average accuracy of 64.66%.

4.3 Number of Features, Patch Size and Whitening

In line with [3], we performed a series of experiments with ASP while varying various dimensions of the feature extraction/classification framework. On the basis of the results of the previous experiments, we applied the multiple input matrix method, a p hyper-parameter value of 0.14 and performed experiments using six different numbers of features (100, 200, 400, 800, 1200 and 1600), three different patch sizes (6×6, 8×8 and 12×12) and with and without whitening. The results from these experiments are graphed in Figure 4 and the results from the best performing configurations of the framework are presented in Table 1 alongside the best results listed in [3].

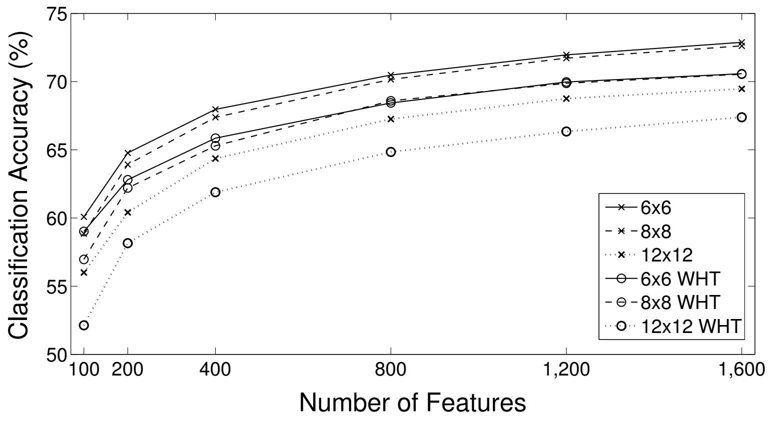

Fig. 4. Average classification accuracy with changes to the number of features. Key represents patch size and WHT indicates data was whitened.

The highest average ASP classification accuracy achieved in these tests was 72.88% using a patch size of 6×6, 1600 features and unwhitened input. As can be seen from the graph in Figure 4, as we increase the feature count, the classification accuracies increase for all patch sizes, with or without whitening. The smallest patch size (6×6) consistently achieves the higher classification accuracy, while the largest (12×12) achieves the lowest. However the gap between the patch size accuracies becomes smaller as the feature count increases, with the difference between 6×6 and 12×12 (unwhitened) patches reducing from 4.075% at 100 features down to 3.415% at 1600 features. However, the difference between 6×6 and 8×8 patches is less pronounced: at 1600 features it is only 0.265% when unwhitened and when whitened becomes negligible after 800 features. In addition, the whitened patches all have lower accuracies than their unwhitened equivalents, with a highest whitened accuracy of 70.58% (using a 6×6 patch size and 1600 features) which is 2.3% less than its unwhitened equivalent.

Table 1. Average classification accuracy using ASP (left) and algorithm results from [3] (right). These results all used 1600 features except where noted.

Algorithm/Patch Size	**Accuracy**	**Algorithm**	**Accuracy**
ASP 6×6	72.9%	Sparse auto-encoder	73.4%
ASP 8×8	72.6%	Sparse RBM	72.4%
ASP 12×12	69.5%	k-means (hard)	68.6%
ASP 6×6 (whitened)	70.6%	k-means (triangle)	77.9%
ASP 8×8 (whitened)	70.5%	k-means (triangle, 4000 features)	79.6%
ASP 12×12 (whitened)	67.6%		

5 Discussion

The main aim of this research is to have developed a version of the ASP algorithm that is competitive with other state-of-the-art feature detection algorithms in the domain of colour image classification. The experimental results show that we have been broadly successful in achieving this aim. Firstly, the two basic innovations we have introduced, *viz* overcompleteness and topography-preserving multichannel connectivity, have both produced significant performance enhancements on the CIFAR data set, and secondly, incorporating these innovations into ASP has produced classification accuracies comparable to the algorithms reported in the Coates *et al.* study.

5.1 Improving ASP

The results in Section 4.1 show that a multiple matrix input configuration can produce a statistically significant improvement in classification accuracy in comparison to a single matrix configuration. This indicates that our topography-preserving strategy is effective on colour image data.

In addition, the results in Section 4.3 show that increasing the number of ASP columns to exceed the number of inputs (i.e. to produce overcomplete encodings) has a strong positive effect on classification accuracy. This is illustrated in Figure 4, where increasing the feature count consistently improves the performance on all configurations of the extraction/classification framework. Although the improvements begin to taper off as the feature count increases, we still see an increase between 1200 and 1600 features, which (when using a patch size of 6×6) have an overcomplete ratio of 11.11 and 14.81, respectively. This ability of ASP to continue to encode useful features of natural images at this level of overcompleteness is desirable, particularly given that V1 is believed to have an overcomplete ratio of over 100 [1].

The results in Figure 4 also show that a patch size of 6×6 consistently produces higher classification accuracies on unwhitened input. However, the distinction between the *whitened* 6×6 and 8×8 patches becomes unclear as the number of features reaches and exceeds 800. These results, as well as the generally lower classification accuracies on whitened patches, indicate that ASP is using information that is being discarded in the whitening process to usefully distinguish features in *unwhitened* data.

The graph in Figure 3 shows that increasing the value of the p hyper-parameter, and thereby connecting each column to more data elements, does not necessarily improve performance. Instead an optimum level of sparsity appears around 0.15, after which increasing column connectivity causes column synapses to overlap in such a way as to degrade their discriminatory power. Setting p at 0.15 was also found to work well in previous studies evaluating ASP on character recognition and greyscale images [15, 23], indicating that this setting is robust to different kinds of input.

5.2 Comparing ASP

Table 1 shows that the highest performing algorithm from the experiments in [3] is k-means with triangle activation, a result which the authors describe as "surprising". This algorithm differs from the more commonly used k-means clustering (shown as k-means (hard) in Table 1), by applying a competitive coding scheme where the more distant centroids are set to zero. This coding scheme reflects a more simplistic form of competition compared to ASP's cortically inspired inhibition model. It does, however, illustrate the effectiveness of competition in forming sparse codes, producing a 10% increase in classification accuracy over k-means (hard), which otherwise would have been the least effective algorithm in the study.

ASP's performance in the current study most closely matches that of the sparse auto-encoder and sparse restricted Boltzmann machine (RBM) reported in [3], with ASP's accuracy of 72.9% sitting midway between the 72.4% of the RBM and the 73.4% of the auto-encoder. These two algorithms have performed well as single layer feature detectors across a broad range of domains [16, 4, 20] and have also been 'stacked' together to form multilayer deep learning networks [24, 11] in a way that is loosely analogous to the incorporation of ASP into the HTM framework. Our results therefore show that ASP's feature detecting abilities are at least competitive with other widely used and comparable feature detecting algorithms.

However, given ASP is outperformed by k-means with triangle activation, we must emphasise that the aim of this study is not to have developed the best possible feature detector for colour images, it is to have developed a version of ASP that is *effective* in discriminating features in colour images. ASP itself is of interest because it represents a biologically plausible model of the kind of feature detection that may be occurring in the neocortex (for example, it uses continuous online Hebbian learning, inter-column inhibition, sparse distributed representations and dynamic creation and destruction of synapses). In addition, ASP is only a component of a larger unified model of the neocortex, whose overall function is not to detect static features of images, but to learn temporal sequences of inputs within a system of mini-columns that are engaged in a hierarchically-structured system of sequence prediction. Consequently, a full evaluation of ASP can only occur within the context of a complete HTM implementation. In the current research, we are instead interested in developing the *best possible* spatial pooler within the constraints and requirements laid down by the overall HTM system. To this end, our basic contribution is to have improved the original version of ASP so that it can now more effectively handle the dimensionality of colour data.

6 Conclusions

The primary contributions of this paper are as follows:

1. We have shown that maintaining the topographic relationship of colour image data is an important factor in the classification performance of ASP and we have provided a simple but generalisable approach to achieve this using the multiple input matrices method.
2. We have presented a suitable way of producing overcomplete ASP feature sets and have demonstrated that these feature sets improve recognition classification as the degree of overcompleteness increases.
3. We have shown, in the domain of colour image classification, that using ASP as a single layer feature detector is competitive with the performance of both sparse auto-encoders and sparse restricted Boltzmann machines.

In future work we plan to integrate ASP with a temporal pooler in a hierarchy of regions and investigate how feedback between regions can be used to improve the system's ability to predict its own input.

References

1. Barlow, H.B.: The Ferrier lecture, 1980: Critical limiting factors in the design of the eye and visual cortex. Proceedings of the Royal Society of London. Series B. Biological Sciences 212(1186), 1–34 (1981)
2. Clark, A.: Whatever next? predictive brains, situated agents, and the future of cognitive science. Behavioral and Brain Sciences 36(3), 181–204 (2013)
3. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics. pp. 215–223 (2011)
4. Deng, J., Zhang, Z., Marchi, E., Schuller, B.: Sparse autoencoder-based feature transfer learning for speech emotion recognition. In: 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction. pp. 511–516 (2013)
5. Fino, E., Yuste, R.: Dense inhibitory connectivity in neocortex. Neuron 69(6), 1188–1203 (2011)
6. Friston, K.: The free-energy principle: a unified brain theory? Nature Reviews Neuroscience 11(2), 127–138 (2010)
7. Hawkins, J., Ahmad, S., Dubinsky, D.: Hierarchical temporal memory including HTM cortical learning algorithms. Tech. rep., Numenta Inc, Palto Alto (2011)
8. Hawkins, J., Blakeslee, S.: On Intelligence. Henry Holt, New York (2004)

9. Hawkins, J., George, D.: Hierarchical temporal memory: Concepts, theory and terminology. Tech. rep., Numenta Inc, Palto Alto (2006)
10. Hebb, D.O.: The organization of behavior: A neuropsychological theory. Wiley, New York (1949)
11. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation 18(7), 1527–1554 (2006)
12. Holmgren, C., Harkany, T., Svennenfors, B., Zilberter, Y.: Pyramidal cell communication within local networks in layer 2/3 of rat neocortex. The Journal of Physiology 551(1), 139–153 (2003)
13. Isaacson, J.S., Scanziani, M.: How inhibition shapes cortical activity. Neuron 72(2), 231–243 (2011)
14. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Computer Science Department, University of Toronto, Toronto (2009)
15. Main, L., Cowley, B., Kneller, A., Thornton, J.: Evaluating sparse codes on handwritten digits. In: AI 2013: Advances in Artificial Intelligence. pp. 396–407 (2013)
16. Malkin, R.G., Waibel, A.: Classifying user environment for mobile applications using linear autoencoding of ambient audio. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 5, pp. 509–512 (2005)
17. Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., Wu, C.: Interneurons of the neocortical inhibitory system. Nature Reviews Neuroscience 5(10), 793–807 (2004)
18. Mountcastle, V.B.: The columnar organization of the neocortex. Brain 120(4), 701–722 (1997)
19. Olshausen, B.A.: Principles of image representation in visual cortex. In: The Visual Neurosciences, pp. 1603–1615. MIT Press, Cambridge, Mass (2003)
20. Teh, Y.W., Hinton, G.E.: Rate-coded restricted Boltzmann machines for face recognition. Advances in Neural Information Processing Systems pp. 908–914 (2001)
21. Thomson, A.M., Lamy, C.: Functional maps of neocortical local circuitry. Frontiers in Neuroscience 1(1), 19–42 (2007)
22. Thornton, J., Main, L., Srbic, A.: Fixed frame temporal pooling. In: AI 2012: Advances in Artificial Intelligence. pp. 707–718 (2012)
23. Thornton, J., Srbic, A.: Spatial pooling for greyscale images. International Journal of Machine Learning and Cybernetics 4(3), 207–216 (2013)
24. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. pp. 1096–1103 (2008)