Evaluating Sparse Codes on Handwritten Digits

Linda Main, Benjamin Cowley, Adam Kneller, and John Thornton

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

Abstract. Sparse coding of visual information has been of interest to the neuroscientific community for many decades and it is widely recognised that sparse codes should exhibit a high degree of statistical independence, typically measured by the kurtosis of the response distributions. In this paper we extend work on the hierarchical temporal memory model by studying the suitability of the augmented spatial pooling (ASP) sparse coding algorithm in comparison with independent component analysis (ICA) when applied to the recognition of handwritten digits. We present an extension to the ASP algorithm that forms synaptic receptive fields located closer to their respective columns and show that this produces lower Naïve Bayes classification errors than both ICA and the original ASP algorithm. In evaluating kurtosis as a predictor of classification performance, we also show that additional measures of dispersion and mutual information are needed to reliably distinguish between competing approaches.

1 Introduction

Investigating the statistical properties of sparse image representations has been a focus of computational neuroscience research for several decades. The motivating hypothesis is that the mammalian neocortex forms models of the environment based on the statistics of visual sensory experience. Sparse coding approaches have been used to demonstrate the neural selectivity of V1 simple cells [16], the neural responses at later stages of visual processing [1], and can be plausibly implemented within the hierarchically structured layers of visual cortex [7].

In previous work [19] it was shown that an augmented version of the spatial pooler algorithm proposed in [6] exhibits desirable properties of sparsity and is a useful feature detector when applied to temporal sequences of images. In the current paper we further investigate this *augmented spatial pooler* (ASP) by comparing the statistical independence of ASP sparse codes with codes generated using *independent component analysis* (ICA). We chose ICA as a comparison algorithm as it is explicitly designed to extract statistically independent signals [9] and is also used in several state-of-art unsupervised feature detection algorithms (e.g. [11]). In order to characterise the nature of the statistical independence, we compare ASP and ICA sparse codes using several statistical measures of sparseness, e.g. kurtosis, dispersion and mutual information. Our focus on statistical

independence is motivated by the assumption that codes with greater statistical independence will better represent the independent causes in the world that generate the raw input of the coding process. We test this assumption by comparing the performance of the two approaches using a *Naïve Bayes classifier*. Our choice of Naïve Bayes is motivated by the expectation that sparse codes with greater statistical independence will produce a better Naïve Bayes classification, i.e. because Naïve Bayes classification itself assumes independence.

The ASP algorithm is one component of the *Hierarchical Temporal Memory* (HTM) model of the neocortex proposed and developed over the last eight years by Jeff Hawkins and various collaborators [6, 7]. The HTM model is a synthesis of general concepts of neocortical function and morphology, and is based on the view that the uniform appearance of the neocortex signifies a similarly uniform functional process [7]. The model is structured as a hierarchical network of identical processing units comprising two functions: a spatial pooler (SP) and a temporal pooler (TP), which cooperate to encode sensory input into temporal sequences [6]. The poolers are modelled as collections of columns, where each column comprises a set of neurons and their associated dendrites and synapses. This columnar structure is modelled on the cortical mini-column, which Mountcastle has described as a basic functional unit within the neocortex [15]. In the spatial pooling function, the columns become active or inactive through inter-column competition and inhibition, which allows HTMs to self organise in direct response to the input while producing sparse distributed representations. The TP then makes Bayesian-like inferences about the temporal ordering of the codes produced by the SP which are fed back into the hierarchy in the form of predictions. This predictive function of the TP distinguishes HTMs from other hierarchical Bayesian models, e.g. [13]. Furthermore, HTM's flexibility and efficiency in processing temporal sequences makes it suitable for online data streams, unlike other sparse coding approaches such as ICA [10].

The current research extends existing work on evaluating the HTM model by studying the relative suitability of the ASP algorithm in comparison with ICA in the domain of handwritten digit recognition. Previous work [20] compared kurtosis measures for ASP and ICA on natural images on the assumption that higher kurtosis would equate to greater statistical independence and hence would produce better feature encoders. Here we question the suitability of kurtosis as the sole indicator of the performance of a feature detector and investigate alternate measures of dispersion and mutual information, while using Naïve Bayes as a measure of relative performance. We also introduce an extension to the ASP algorithm that clusters synapses more closely to their respective columns and show how this can produce a lower Naïve Bayes classification error.

In the remainder of the paper we provide details of the ASP algorithm and our own extensions, followed by a brief description of the FastICA algorithm we used for comparison purposes. We then provide details of the statistical measures used to evaluate the ASP and ICA sparse codes and present the results of our experiments on the well known MNIST handwritten digits dataset. Finally the significance of these results are analysed and discussed.

2 Augmented Spatial Pooling

The basic functional unit of the HTM poolers is a *column* which consists of a collection of cells and dendrites that form synaptic connections with other columns and with an input layer. The HTM temporal pooler connects individual cells within a column to cells within other columns and the spatial pooler connects an entire column to the elements of an input layer. As the paper is concerned with SP sparse codes, we shall not consider the temporal pooler further. Instead, we shall treat a column as a single *coding unit* (CU) with dendrites that only synapse with the individual elements of an input layer.

Unlike more traditional neural networks, the SP connections are not multiplicatively weighted to determine the strength of the input signal. Instead, the synapses are *potential* and assigned a *permanence value* which indicates whether the synapse is *connected* (i.e. active), or potential and inactive. It is only when the permanence value of a synapse passes a certain threshold that the synapse becomes connected and the value of the input to which it is connected is then passed on by the dendrite. The activity level for a column is then calculated as the sum of the input from all its connected synapses.

Each column is positioned above a single pixel of the input image and no pixel has more than one column above it. A set of dendrites from each column is mapped to a subset of the pixels of the image, where each dendrite forms a single synaptic connection to a single pixel. This topographic layout allows for the calculation of the Euclidean distances from a column to each of its potential synapses. The column's list of potential synapses is then ordered by these distances, and this ordering is used when determining the initial synapse permanence values. The area that bounds a column's connected synapses is termed the column's *receptive field*, and the mean receptive field size of all columns is the size of the *inhibition area*. The sparse encoding of an input pattern is produced via a process of competition whereby the more active columns within a given inhibition area inhibit (or switch off) their less active neighbours.

Learning in the spatial pooler is based on how well the column synapses match (or overlap) the input to which they are connected. This is achieved by modifying the synapse permanence values of the columns which win the inhibition competition – synapses connected to active input have their permanence values increased, while synapses connected to inactive input have their permanence values decreased. In addition, columns that fail to reach a minimum average activation threshold are able to competitively form new synapses with the input (for full details see [19]).

Initialising the Spatial Pooler Synapses: The process of initialising the columns and their synapses is presented in Algorithm 1. The first step is to randomly select a percentage of the image pixels to which potential synapses will be mapped according to the value of $P(potential)$ (in the current implementation, this is set to 0.1). The next step is to probabilistically set the perm(s) permanence values for each of the potential synapses for all columns according to the

original ASP algorithm [20] or according to two modified approaches (ASP+M and ASP+G) detailed below.

The last step of initialisation is to calculate each column's receptive field area (line 13) and the inhibition area size (line 15) to be used by all columns during the inhibition competition. After each training instance is presented to the system the receptive fields of all columns and the inhibition area are recalculated.

Algorithm 1 initialiseColumns($columns, method, m$)

```
 1: for each column c in columns do
 2:   for each synapse s in column c do
 3:     if random(0,1) ≤ P(potential) then
 4:       if method is ASP then
 5:         perm(s) = threshold + (random(0.0, 0.1) − distance(s))
 6:       else if method is ASP+M then
 7:         perm(s) = threshold + (random(0.0, 0.1) − (distance(s) × m))
 8:       else if method is ASP+G then
 9:         perm(s) = gaussianPDF(distance(s) × m, 0, σ) + random(-0.05, 0.05)
10:       end if
11:     end if
12:   end for
13:   totalReceptiveFieldAreas += calculateReceptiveFieldArea(c)
14: end for
15: inhibitionArea = totalReceptiveFieldAreas / numCols
```

ASP+M: In the HTM specifications [6] initial synapse permanence values are randomly set to within a small range of the permanence *threshold* and are linearly biased such that synapses which are closer to their column are more likely to have an initial permanence value at or above the threshold. In the original ASP algorithm [19] the synapse permanence values have a potential range of 0 to 1 with the threshold set at 0.2 and initial permanences bound to be within 0.1 of the threshold. We use these values in our current study (see lines 4–5).

By way of extension, we also experimented with a multiplier m that increases the range of possible initial permanence values, such that they are no longer within the 0.1 bound. This multiplier increases the probability of synapses close to their column having permanence values above the threshold, and decreases the probability for those further from their column. We name this method ASP+M (see lines 6–7).

For both ASP and ASP+M, we standardise the distance(s) measure for each synapse to range between 0.0 and 0.1. In ASP this distance is subtracted from a random number that also ranges from 0.0 to 0.1 and the result is added to the preset *threshold* value of 0.2. In ASP+M, we simply multiply the normalised distance value by a value m and proceed as for ASP (see line 7).

ASP+G: The second extension to ASP replaces the linear bias with a Gaussian distribution centred on each column, where synapses furthest from the column

are least likely to have their permanence values set above the connection threshold. The initial permanence values are then randomly selected from within a boundary of 0.05 of the Gaussian probability density function. We name this method ASP+G (see lines 8–9).

ASP+G converts a synapse's distance(s) from its column into a number of standard deviations from the mean of a Gaussian, such that a column's most distant synapse is set to m standard deviations (in the experimental study we set $m = 5$). We then calculate the probability of this distance using a Gaussian probability density function (PDF) with mean 0 and standard deviation σ. The synapse's *permanence* value equals this probability plus a random noise value between ± 0.05 (see line 9).

3 Independent Component Analysis

Our experimental study compares ASP sparse codes with codes generated using independent component analysis (ICA). ICA belongs to the class of *blind source separation* (BSS) methods aimed at separating data into maximally independent features. It is based on the assumption that the sources are non-Gaussian and statistically independent [17]. However, particularly in the case of natural images, the independence assumption has been criticised and has led to the development of *independent subspace analysis* (ISA) which, in addition to performing ICA, also attempts to find independent *groups* of features [9]. In order to equitably compare ASP codes with the higher order codes learned by ISA, a hierarchical implementation of spatial pooling would also be required. However, such a hierarchical implementation runs the risk of obfuscating the underlying performance of the two algorithms. In contrast, the lower order functionality of ICA is more directly comparable to a single layer ASP and allows for unmodified statistical measurements. For these reasons we have not included ISA in the study.

ICA may be considered a multivariate, parallel version of projection pursuit which seeks to maximise the kurtosis, or alternatively minimise the mutual information, of separated signals. Maximising kurtosis is based on the assumption that a mixed signal, which is often Gaussian in nature, is composed of independent non-Gaussian signals. Further, ICA assumes that any Gaussian signal may contain noise, which is first separated from the mixed signal by applying *principle component analysis* (PCA) [17]. This may result in reducing the dimensions of the mixed signal, and consequently, ICA will extract fewer signals than suggested by the dimensions of the raw input.

The Matlab-based FastICA implementation of ICA, which we use in this study, employs the Kullback-Leibler divergence as a measure of the distance between an extracted signal and a Gaussian signal, and seeks to maximise that distance in the gradient descent search [10]. As FastICA explicitly maximises non-Gaussianity (i.e. kurtosis) during the learning process, we expect the sparse codes it produces will display high kurtosis.

4 Statistical Properties of Sparse Codes

In this study we are interested in the statistical properties of ASP and ICA sparse codes and apply a series of measurements to enable a quantitative comparison between them. For the purpose of these comparisons, we define a *coding unit* (CU) as a column in ASP and a basis function in ICA. For both algorithms, the CUs are the constituents of the sparse responses to the image data.

Willmore, Mazer and Gallant [21] describe a range of statistical measures used to characterise sparse codes and note that different studies used similar terminology but in different contexts, making the comparison of the models and their sparseness measurements problematic. To resolve some of the confusion, they clearly define the different interpretations of sparseness and present a taxonomy of these concepts. The first classification they describe distinguishes *overall activity* from the *shapes of response distributions*. Overall activity measures the mean firing rates of neurons and is intuitively suitable for characterising the behaviour of biological neurons rather than computational models of neurons. As the sparse CUs of ASP and ICA are theoretical *collections* of neurons, we do not attempt similar measurements in this study. Instead we investigate the shapes of response distributions which can be measured in two distinct dimensions: *lifetime* measures that characterise the response distribution of single CUs to many images and *population* measures that characterise the response distribution of an entire collection of CUs to a single image instance. The following sections detail several such measures that we will use in the experimental comparison:

4.1 Lifetime Measures

Lifetime Sparseness: A response distribution where CUs are primarily inactive but occasionally respond strongly is defined as having high lifetime sparseness. A response of this kind may be characterised by measuring the kurtosis of the response distribution. To compare the lifetime sparseness of ASP and ICA sparse codes, we calculate the average kurtosis of each CU's responses to an entire set of input images using the method in [22].

Maximisation of Information: Willmore et al. [21] state that if the visual cortex maximises information, then the response distribution should be exponential in shape. In this respect we have fitted an exponential model, of the form ae^{bx}, to the response distribution of each CU of ASP and ICA. We use the goodness of fit (measured by the root mean square error, RMSE) of the model to characterise how well the CU's response distribution matches the exponential distribution. A close fit will result in a low RMSE and indicate that the sparse codes of the distribution have a high degree of information.

4.2 Population Measures

Population Sparseness: A sparse code where the population response distribution to a single image has high kurtosis is referred to as having high population

sparseness, i.e. only a small proportion of the CUs is active at any given time. We calculate the population kurtosis of a single image using the method in [22].

Mutual Information Minimisation: Given two random variables, X and Y, the *mutual information*, $I(X;Y)$, between these variables is:

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \tag{1}$$

where $P_X(x)$ and $P_Y(y)$ are the probability mass functions of X and Y respectively, and $P_{XY}(x,y)$ is their joint probability mass function. If we consider a single CU to be a random variable whose distribution of responses to a set of M images constitutes its probability mass function, then we may calculate the average mutual information between all pairs of CUs as follows:

$$averageMutualInformation = \frac{2}{(N-1)N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} I(i;j) \tag{2}$$

where $I(i;j)$ is the mutual information between CUs i and j computed as above, and $\frac{2}{(N-1)N}$ is the number of distinct pairs possible from N CUs. This gives us a measure of the average amount of information that is shared between CUs. High mutual information values indicate that the sparse codes have a high level of redundancy, whereas an efficient system will have low redundancy producing a low mutual information value [3].

Redundancy Minimisation: As a second measure of redundancy, we consider the degree of load sharing by counting the number of sparse codes in which the CUs participate. Following [22], for each image we set a threshold value equal to the standard deviation of the response distribution for that image. CUs whose response magnitude for that image is greater than the threshold value are considered 'on' and CUs with response magnitudes less than the threshold value are considered 'off'. The binarised response of CU c to image i is then calculated as:

$$binarisedActivity_i^c = \begin{cases} 1 & \text{if } a_i^c > \sigma_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where a_i^c is the response magnitude of CU c to image i, and σ_i is the standard deviation of the CU population's response distribution to image i. The participation level of a CU c is then the total count of the sparse codes in which it is considered 'on'.

Dispersion: The response of a CU to an image will vary from strongly positive, where an image matches the CU's receptive field, through zero where the image is orthogonal to the receptive field, to strongly negative where the image is the inverse of the receptive field. If an image set has a high variance along the axis of the CU's receptive field then the responses of the unit over the set will show

a high variance between the positive and negative extremes. This indicates that the CU's receptive field is well matched to some of the variations in the image set and the CU is therefore appropriate for encoding differences between members of this set [23]. Conversely, a low variance indicates that the variations in the image set are largely orthogonal to the CU's receptive field and that the CU is not appropriate for encoding variations in the image set.

By comparing the response variances of all CUs in a set, we can obtain a measure of how evenly the coding is *dispersed*. Willmore, Watters and Tolhurst [23] suggest visualising and quantifying these differences using scree plots. After binarising the ASP and ICA CU activities we calculate and normalise the variance of all units such that the maximum is 1. We plot the variances in rank order, highest to lowest, and use the area under the curve as a measure of dispersion. A low area indicates that there are a few high variance CUs that are encoding the majority of the variations in the image set, i.e. the coding is concentrated, not dispersed. Conversely, a large area shows there are many high variance CUs sharing the encoding and the resulting codes have greater dispersion.

5 Experimental Evaluation and Discussion

As earlier spatial pooler versions have been tested on character recognition problems (e.g. [5]) we chose to test our modified spatial poolers, ASP+M and ASP+G, and ICA on the well known MNIST handwritten digits dataset [12]. The set comprises 70,000 handwritten images (28×28 pixels) of the digits 0–9, split into 60,000 training images and 10,000 test images. ASP and FastICA use random seeds, so we executed all experiments five times using different seeds, and report the average of these executions. All tests were conducted on a Dell Optiplex 990 3.10 GHz Intel Core i5 processor with 16 Gb of 1333 MHz DDR3 RAM running Windows 7 Enterprise v. 6.1 and Matlab v. 7.12 (R2011a). For classification we used the Matlab Statistics Toolbox Naïve Bayes classifier.

Varying Initial Receptive Field Size: We tested our modified ASP algorithms on a range of settings in order to investigate the effect of the different methods for initialising ASP's synapse permanence values. ASP+M was tested using multiplier values 0–6 with 4 giving the highest classification. ASP+G was tested by altering the standard deviation of the Gaussian distribution from 0.1 to 1.8 in steps of 0.1, with 0.9 producing the best classification result. Table 1 summarises the mean Naïve Bayes classification accuracies, and population and lifetime kurtosis of the response distributions for ASP, ASP+M and ASP+G. Setting the ASP+M multiplier to 0 is equivalent to removing the linear bias and setting synapse permanence values randomly without respect to their distance from their column. Results for this setting are included in Table 1 as ASP− and show that randomly selecting active synapses without any topographic reference to their column produce the worst classification accuracy (82.65%), population kurtosis (38.30) and lifetime kurtosis (49.87).

Table 1: Summary of results for ASP, ASP+M, ASP+G and ICA.

Algorithm	ASP−	ASP	ASP+M	ASP+G	ICA
Accuracy (%)	82.65	84.39	88.62	89.45	86.12
Population Kurtosis	38.30	47.80	73.79	65.48	48.13
Lifetime Kurtosis	49.87	60.85	98.43	83.86	4864.19
Mutual Information	—	—	—	0.0082	0.0063
Goodness of Fit (RMSE)	—	—	—	7.26	1190.10
Dispersion Area	—	—	—	121.06	177.20

The ASP+M multiplier causes active synapses to be more clustered around their column than for ASP, and could explain ASP+M's higher accuracy of 88.62 (compared to ASP's accuracy of 84.39). The more clustered receptive fields of ASP+M (with a mean $1:4.975$ per column ratio of 8 active synapses to 39.80 receptive field pixels) compared to ASP (with a mean $1:11.39$ per column ratio of 13 active synapses to 148.10 receptive field pixels) allows ASP+M to encode more specific features than ASP. Both ASP's population kurtosis (47.80) and lifetime kurtosis (60.85) are lower than ASP+M's (73.79 and 98.43 respectively) indicating that smaller and more densely clustered receptive fields produce codes exhibiting greater sparseness and higher classification accuracies.

In contrast, ASP+G has approximately the same ratio of active synapses to receptive field size as ASP (6 synapses to 61.30 pixels or $1:10.21$), but achieves this ratio using approximately half as many synapses covering a much smaller mean receptive field. While the smaller more clustered receptive fields of ASP+M are sensing more specific features than the larger receptive fields of ASP+G, the denser synaptic connections of the ASP+M columns suggest insufficient information is being sensed to uniquely encode the classes. However, ASP+G's lower population kurtosis of 65.48 and lower lifetime kurtosis of 83.86 compared to ASP+M (the highest of the ASP methods) would lead us to expect ASP+G to have lower accuracy than ASP+M, whereas the reverse is the case. This suggests that kurtosis is not a perfect predictor of accuracy.

Comparison with ICA: Comparing the lifetime kurtosis of ASP+G and ICA (83.86 and 4,864.19 respectively) to their population kurtosis (65.48 for ASP+G and 48.13 for ICA) further supports the argument that using only kurtosis as a performance indicator is inappropriate. The very high lifetime kurtosis of ICA would lead us to expect a correspondingly high accuracy, whereas ASP+G actually outperforms ICA. This anomalous result led us to manually inspect the binarised ICA response distributions, revealing that an average of 10 ICA CUs had only responded to a single image and were therefore acting as *instance detectors* rather than as members of a feature detector. In addition, ICA's response distributions had an average of 36 overly sparse codes (i.e. responses with 5 or less CUs) and were typically encoded by the same CUs within the distri-

bution, which explains the disproportionately high lifetime kurtosis. Response distributions with CU behaviours of this nature are indicative of a model having overfitted to the data.

When we consider that ICA iteratively seeks to extract the *most independent* signal, it is to be expected that it would extract instance detectors when applied to the relatively simple images of handwritten digits. As ICA was designed to perform sparse encoding of natural images, and is now overfitting this simpler dataset, we consider ICA to be a less general sparse encoder than ASP, which competitively encodes a wider range of image types.

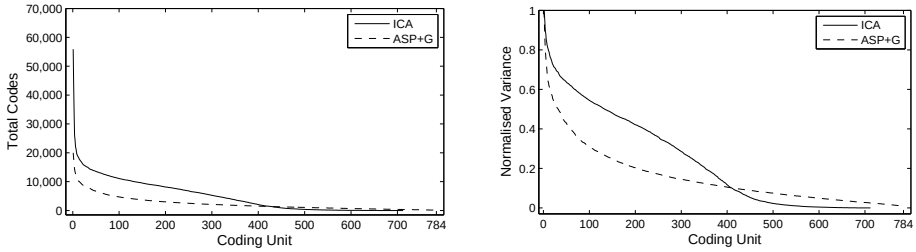

Fig. 1: Left: Coding unit (CU) participation counts for ASP+G and ICA. **Right:** Dispersion area for ASP+G and ICA.

Next, we consider the degree of redundancy by measuring the mutual information shared between CUs. It is reasonable to expect that CUs of an overfitted model will have very low mutual information because they are active in isolation or with very few other CUs. This goes against the theoretical expectation that CUs should share minimal information so as to reduce redundancy. Clearly, some mutual information is needed to ensure the CUs are acting as members of feature detectors and not as instance detectors. We find that the mutual information for ASP+G and ICA (at 0.0082 and 0.0063 respectively) supports this interpretation of the response distributions, when considered *together* with the high lifetime kurtosis and lower accuracy of ICA.

From our alternative dispersion measure using participation counts (see Figure 1), we find further support for overfitting by ICA. The high peak participation for ICA is caused by a mean of 8 CUs which are active in more than 20,000 sparse codes. At the other end of the graph we notice a mean of 13 CUs which *fail to participate in any sparse code*, which indicates that ICA has learned redundant basis functions. The mean activity for ICA is 1,358.53, considerably larger than that of ASP+G (365.16), showing that ASP+G produces more dispersed codes, as encoding is more evenly distributed. This is a result of the underlying ASP algorithm's competitive nature which is explicitly designed to disperse the encoding. In contrast, the dispersion measure of [23] (graphed in Figure 1) indicates the ICA response distributions (177.20) are more dispersed than ASP+G (121.06).

When we measured the degree of information retained in the codes (by fitting an exponential curve to the response distribution and calculating the RMSE of the fit), the RMSE for ASP+G was 7.26 compared to 1,190.10 for ICA. This indicates ASP+G is encoding more salient class features, whereas ICA's high RMSE demonstrates its loss of class specific information as it encodes instances.

Finally, a t-test, with p-value of 0.05, performed on ASP+G and ICA classification accuracies rejects the null hypothesis, indicating that the difference in mean accuracies between ASP+G and ICA is statistically significant.

As ASP runs 8.36 times faster than ICA (with a mean convergence time of 299 seconds compared with 2,582 seconds for ICA), and is simpler to implement than the complex gradient descent approach of ICA (for which the neural mechanisms have yet to be elucidated), we consider ASP a more plausible model of feature detection activity within the mammalian neocortex.

6 Conclusions

In conclusion, the paper has made three main contributions:
1. We have given grounds for believing that ASP is a *better* sparse encoder than ICA in the domain of character recognition. This is firstly because ASP is faster (i.e. more efficient) than ICA at generating codes, which also suggests the ASP strategy is more biologically plausible. Secondly, ASP produces better Naïve Bayes classification accuracy, suggesting that, in practice, ASP is a more useful feature detector than ICA for domains such as character recognition, where the ICA basis functions can degenerate into simple instance detectors.
2. We have shown that kurtosis on its own is not necessarily the best way to measure the statistical independence of sparse codes and that additional measures of dispersion and mutual information are needed to give a reliable picture of the potential performance of an encoder.
3. We have shown there is an important relationship between the synapse initialisation strategy used for ASP and the subsequent classification accuracy of the generated codes. Specifically we have shown that a Gaussian distribution that strongly clusters synapses within a short distance of their associated column is better than the alternative linear and linear multiplicative strategies tested.

In future work we plan to investigate both class-conditional ICA and ISA approaches to character recognition by developing analogous class-conditional and hierarchical ASP algorithms. We also plan to extend our work to the domain natural images and, upon integration of a TP, to temporal data such as sound.

References

1. Carlson, E.T., Rasquinha, R.J., Zhang, K., Connor, C.E.: A sparse object coding scheme in area v4. Current Biology 21(4), 288–293 (2011)
2. Felleman, D.J., Van Essen, D.C.: Distributed hierarchical processing in the primate cerebral cortex. Cerebral cortex 1(1), 1–47 (1991)

3. Friston, K.: Learning and inference in the brain. Neural Networks 16(9), 1325–1352 (2003)
4. Fu, M., Yu, X., Lu, J., Zuo, Y.: Repetitive motor learning induces coordinated formation of clustered dendritic spines in vivo. Nature 483(7387), 92–95 (2012)
5. George, D., Hawkins, J.: A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN-05). pp. 1812–1817 (2005)
6. Hawkins, J., Ahmad, S., Dubinsky, D.: Hierarchical temporal memory including HTM cortical learning algorithms. Tech. rep., Numenta, Inc, Palto Alto (2011), https://www.groksolutions.com/technology.html#cla-whitepaper
7. Hawkins, J., Blakeslee, S.: On intelligence. Henry Holt, New York (2004)
8. 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
9. Hyvärinen, A., Hurri, J., Hoyer, P.: Natural Image Statistics: A probabilistic approach to early computational vision. Springer-Verlag New York Inc (2009)
10. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Components Analysis. John Wiley and Sons, Inc., New York (2001)
11. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 3361–3368 (2011)
12. LeCun, Y., Cortes, C.: MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist (1998)
13. Lee, T.S., Mumford, D.: Hierarchical Bayesian inference in visual cortex. Journal of the Optical Society of America A 20(7), 1434–1448 (2003)
14. Malone, B.J., Kumar, V.R., Ringach, D.L.: Dynamics of receptive field size in primary visual cortex. Journal of neurophysiology 97(1), 407–414 (2007)
15. Mountcastle, V.B.: Introduction to the special issue on computation in cortical columns. Cerebral Cortex 13(1), 2–4 (2003)
16. Olshausen, B.A., et al.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583), 607–609 (1996)
17. Stone, J.V.: Independent component analysis. Wiley Online Library (2004)
18. Thornton, J., Main, L., Srbic, A.: Fixed frame temporal pooling. In: AI 2012: Advances in Artificial Intelligence, pp. 707–718 (2012)
19. Thornton, J., Srbic, A.: Spatial pooling for greyscale images. International Journal of Machine Learning and Cybernetics 4, 207–216 (2013)
20. Thornton, J., Srbic, A., Main, L., Chitsaz, M.: Augmented spatial pooling. In: AI 2011: Advances in Artificial Intelligence, pp. 261–270. Springer (2011)
21. Willmore, B.D., Mazer, J.A., Gallant, J.L.: Sparse coding in striate and extrastriate visual cortex. Journal of neurophysiology 105(6), 2907–2919 (2011)
22. Willmore, B., Tolhurst, D.J.: Characterizing the sparseness of neural codes. Network: Computation in Neural Systems 12(3), 255–270 (2001)
23. Willmore, B., Watters, P.A., Tolhurst, D.J.: A comparison of natural-image-based models of simple-cell coding. Perception-London 29(9), 1017–1040 (2000)