Stable Sparse Encoding for Predictive Processing

Linda Main
Institute for Integrated and Intelligent Systems
School of ICT, Griffith University
Queensland, Australia
Email: linda.main@griffithuni.edu.au

John Thornton
Institute for Integrated and Intelligent Systems
School of ICT, Griffith University
Queensland, Australia
Email: j.thornton@griffith.edu.au

***Abstract*—Systems which fall within the domain of the Hierarchical Predictive Coding principle and which have adopted prediction as their primary goal, are heavily reliant on stable sparse coding of sensory input. Furthermore, such systems will require their spatial coding function to be adaptive and able to reform to reflect changes within the environment. These properties of stability and adaptiveness should be emergent properties of the spatial coding system and not reliant on additional control mechanisms. Hierarchical Temporal Memory is a cortically inspired model that encapsulates both sparse coding and temporal processing functions. We present an investigation into the stability and adaptiveness of three alternative versions of the spatial pooling function. Our results show that the SP algorithm is able to form stable sparse distributed representations of audio input, while still remaining adaptive to changes within the input data.**

I. Introduction

The predictive processing model is a current and highly influential theory of brain function [1], [2]. It proposes that the neocortex is engaged in predicting incoming sensory information facilitated by the use of error message passing between its layers [3]. By adopting prediction as the primary processing objective, the emphasis in recent research has moved beyond sensory encoding techniques towards investigating temporally-based aspects of cortical function. However, for any temporal processes to be performed, there remains the need for an optimal process of sensory encoding.

The vast complexities of the dynamic environments in which mammals exist demand that their neocortices process incoming information in a timely and efficient manner. Therefore, their cortical systems should be both self-organising and adaptive, and not reliant on external control processes which would impede the immediacy of cortical response to changing external stimuli. Stable sensory encoding, which is also able to adapt to the arrival of new data, is needed to facilitate further processing in the temporal dimension, i.e. to allow prediction-based functions to be optimally performed. Furthermore, the emergent properties of self-organisation and adaptiveness should promote system stability in both the spatial and temporal dimensions so as to maximise survival abilities.

After several decades of research into possible cortical coding strategies, there remains little doubt that the fundamental principle is that of sparse distributed representation (SDR) [4]. A sparse code is one in which only a small percentage of the coding elements are active at any given time. Distribution of the coding elements throughout the coding cohort expands the range of code permutations possible. When employed within a hierarchy, it permits abstraction of input that results in re-useable components which further widen the range of inputs that may be encoded. Following the seminal work of Olshausen and Field [5], sparse coding gained popularity in computational models of perception and learning. Their mathematically formalised approach provided a plausible account of how such encodings may be realised in neocortex and demonstrated its ability to process the complexities of the environment. The SDR principle has been widely demonstrated and applied to a range of sensory input, initially with a focus on visual data [6]. Early work on auditory encoding targeted speech recognition [7] and was then expanded to include bioacoustics, with birdsong being a popular choice in this category [8].

The goal of this study is to observe the stability and investigate the adaptiveness of the cortically inspired machine learning model, Hierarchical Temporal Memory (HTM), when applied to the task of encoding birdsongs. Amongst the various proposals for AGI architectures [9], we consider HTM [10] a promising approach due to its relatively simple structure and functions. In comparison to other models at comparable levels of abstraction, HTM encapsulates more aspects of the neocortex. HTM is fundamentally based on temporal prediction, making it a departure from mainstream AI research where temporal concerns have typically been retro-fitted to established models [11], [12], [13].

HTM is composed of two interdependent functions: Spatial Pooling and Temporal Pooling (SP and TP resp.), using a single structural model. Due to the demand of temporal processing for stable spatial encoding, this study is focussed on the behaviour of the SP during an encoding task. Specifically, we compare two alternative implementations of the SP algorithm: Augmented SP (ASP) and nupicSP, in terms of their stability and adaptiveness. This is the first time a comparison of these algorithms, when applied to encoding auditory data, has been undertaken. In addition, this is an initial, comparative investigation of the stability and adaptiveness of ASP and nupicSP.

The results of our experiments show that ASP is the better performing encoder in terms stability and adaptiveness when applied to birdsong data. We observed that it reaches a stable

equilibrium at low levels of noise, and remains adaptive as more noise is introduced. Lastly, we implement a modification to the algorithm which improves the quality of ASP's output for use in birdsong classification, without undue impact on its stability and adaptiveness.

The next section provides a description of HTM and details of ASP and nupicSP. Our experiments and results are then described, and followed by a section discussing the outcomes. The paper ends with our conclusions drawn from this work and directions for future research.

II. HTM AND SP

Conceived as a high-level abstraction of neocortical structure, HTM provides an alternative synthesis of widely accepted cortical principles. It fits within the free energy formulation of Friston [14], as its primary goal is to learn and predict sequences in incoming sensory information which is best achieved by minimising unexpected interaction with the environment. What sets HTM apart from other implementations in the Hierarchical Predictive Coding (HPC) class is its unique model structure which makes use of hierarchically arranged networks of artificial *mini-columns* that are focussed on learning the *sequences* received from lower regions assisted by feedback from higher regions.

Following its initial description by Hawkins and Blakesley [15], the release of technical details for HTM [10] provided pseudocode for implementing the two processing functions of the model: Spatial Pooling (SP) and Temporal Pooling (TP). An open source project was subsequently made available to the research community under the name "nupic"[1]. It is from this project that we sourced the nupicSP implementation for this study. ASP is an extension to the SP algorithm which has been previously investigated [25], and which we used as the base for our algorithmic extension in this work.

Interest in HTM has steadily grown, with various researchers now turning their attention to the model. The history of HTM development and review of some of the HTM research are discussed in the article by Awad and Khanna [16]. Currently, much of the work is still focussed on the SP [17], [18], with several mathematical formulations of SP and its computational properties recently being published [19], [20], [21]. However, some investigations have looked toward incorporating feedback mechanisms [22], while others are finding novel applications for HTM [23], [24].

1) HTM Structure: The distinguishing features of HTM's structure are based on its use of a more sophisticated model of cortical cells than is typically used in neural network (NN) models, where only a single type of connection is commonly used. In contrast, an HTM neuron incorporates *two distinctly different* types of connections to represent proximal and distal dendrites (refer Figure 1). Columns of HTM cells are formed to represent cortical mini-columns. These columns are the fundamental processing unit of the model and are a unique aspect of HTM structure and function.

[1]nupic. https://github.com/numenta/nupic

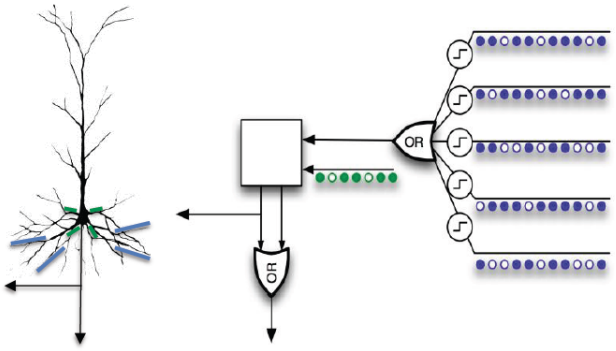

Fig. 1: Left: Biological cell. Right: HTM cell showing the TP distal dendrites in blue, and the SP proximal dendrites in green. Image taken from [10].

Another key difference in the structure of HTM is its use of *regions*, which are grouped arrangements of HTM columns. Like other approaches which have turned their attention to hierarchical modelling, HTM also uses this architecture but with the key difference that any given layer in the hierarchy may be formed by multiple regions. According to HTM theory, these regions would permit processing of different types of sensory or contextual input which, when presented to the next higher layer in the hierarchy, facilitate more refined abstraction.

2) Temporal Pooling: HTM's fundamental principle is based on the premise that learning is essentially a task of sequence identification and prediction. This function is performed by the TP, which makes use of the distal dendrites of HTM cells. These dendrites are composed of *segments* which make individual lateral connections to cells in different columns of the region as the system processes the feed-forward temporal stream. Based on the state of the cells (i.e. inactive, active or predicted) in the temporal context, these dendritic connections permit predictions of further input. As this study is focussed on the SP, we refer the interested reader to the HTM literature [10] for more details on the TP function.

A. Spatial Pooling

Within any region of the HTM hierarchy, the first operation to be performed is spatial processing of feed-forward input. The SP is employed in this task with the singular goal of forming SDRs of input such that the TP is best enabled to perform its prediction objectives. Accordingly, the codes produced by the SP should be temporally consistent; an output which would naturally occur from an adapted and stable function.

In contrast to the TP which makes use of the distal dendrites of HTM cells, the SP uses the proximal dendrites. These connections are not segmented, instead forming single synapses to input in a manner similar to other NN models. However, the distinguishing difference of HTM synapses is the use of their *permanence* value together with the system's permanence *threshold* parameter. Synapses with permanences

greater than the threshold are deemed *connected*, while those with permanences lower than the threshold are considered *potential* and *inactive*. An *active* synapse is one that is connected and synapses to non-zero input. It is only connected and active synapses that transmit their (unmodified) input. During the learning phase of SP, synapse permanences are either positively or negatively reinforced, thereby fulfilling Hebbian learning principles. Mnatzaganian et al. [19] describe the synapse permanence as a probability of connection, which differs from classical NNs where the connection weight is used to alter the signal being transmitted.

SP output is the learned pattern of *columnar activations* for any given input. In order to determine a column's activation level, the input passed by the active synapses of all cells of the column is summed (termed the column *overlap*) and then multiplied by the column's *boost* factor. Boosting is used to ensure all columns meet minimum participation levels, and together with an *inhibition* process, ensure distribution of the sparse codes. As the synapses of each cell within a column contribute to the total activation of the column, the model structure makes no distinction between the individual cells in the context of the SP. Accordingly, each SP column is represented simply by the set of proximal dendrites for all its cells. The internal processes of SP are performed in four sequential stages:

1) **Calculate overlap** determines the activation level of the columns (described above).
2) **Inhibit columns** determines the active and inactive columns (described below).
3) **Synaptic learning** increments synapse permanences of active columns and decrements synapse permanences of inactive columns.
4) **Column boosting** increases the boost factor of inactive columns.

1) Inhibition: Within the SP, the use of inhibition between the columns ensures distribution of the coding columns in the output SDRs. This capacity of the system is self-organising and allows it to adjust itself to the structure of the input data.

In order to determine which columns will participate in the inhibition process, an inhibition area (termed the *neighbourhood*) is calculated first. For each column of the SP, a *receptive field* is calculated as the area of the input spanned by its connected synapses. Taking the mean of all columns' receptive fields produces the size of the neighbourhood, which is then used by each column during inhibition. Algorithm 1 describes the inhibition process.

The *desiredActivity* is a system parameter that determines the number of active columns within each column's neighbourhood. In order for a column to be set active, there must be less than the desired number of active columns amongst that column's neighbours. When counting the number of existing active columns within a neighbourhood, a neighbouring column's overlap must be greater than the overlap of the column being processed. If the number of active columns within a column's neighbourhood meets the desired activity level, the column is effectively inhibited by failing to be set active. The *desiredActivity* parameter has a direct impact on the density of coding columns in the learned SDRs. For example, if this value is set high, more columns will pass the inhibition test and result in the SDR having a denser proportion of active columns. However, if this value is set lower, less columns will pass the inhibition test and therefore undergo further synaptic learning and column boosting. This parameter is one of the few system variables that requires tuning for any given dataset to ensure the desired sparsity of the SDRs produced by the system.

Algorithm 1 Inhibit Columns in ASP

for each column c **do**
  $active(c) =$ false and $activitySum = 0$
  for each neighbour n of c within $neighbourhood$ **do**
    if $overlap(n) > overlap(c)$ **then**
      $activitySum = activitySum + 1$
  if $activitySum < desiredActivity$ **then**
    $active(c) =$ true

2) ASP vs nupicSP: The most significant difference in ASP when compared to nupicSP is the introduction of a more robust learning strategy aimed at addressing the premise that the system should attain equilibrium in order to produce stable SDRs which are needed to facilitate sequence learning. The SP boosting process often fails to sufficiently alter the pattern of connected synapses, despite elevating inactive columns into activity. Consequently, nupicSP can fail to converge, especially on complex (high entropy) input. In contrast, ASP adopts a more conservative method for assigning new synapses to columns while allowing more boosting to occur which results in better convergence properties of the system [25].

The second significant difference between the two SP algorithms is the limitation of nupicSP being able to process *only* binary-valued input, whereas ASP was altered to permit handling of grey-scale images. ASP's ability to process real-valued input allows both a wider range and more biologically plausible sensory input, without the need to perform a data conversion to binary-values.

Further development of ASP provided an alternative method of initialising synapse permanences, which was shown to produce lower classification error when applied to handwritten digits [26]. Where nupicSP uses an unweighted random selection of initial synapse locations and permanences, ASP uses a Gaussian-biased random selection (centred on each column) that results in initial connected synapses more likely to be located closer to their respective column.

3) Restricted Neighbourhood ASP (rnASP): As described above, the effect of the *desiredActivity* parameter during the inhibit columns stage of SP processing is to control the number of columns that will win the inhibition competition. Columns that fail to win the competition will undergo further learning in subsequent stages of the SP process. In this study we introduce an extension to the inhibit columns stage by adopting an additional *outputActivity* parameter that allows us to set different levels of activity during the learning and output

processes. By specifying less activity during learning (which may be considered as restricting the activity of the neighbourhood), we are able to cause more columns to undergo further learning. However, as the activity parameter also has a direct impact on the density of the final output SDRs, the additional *outputActivity* parameter is required to ensure that the final learned codes meet the desired sparsity levels. This addition is detailed in Algorithm 2, and the experimental results from this extension are presented in the next section.

Algorithm 2 Inhibit Columns in rnASP

```
for each column c do
    active(c) = false and activitySum = 0
    for each neighbour n of c within neighbourhood do
        if overlap(n) > overlap(c) then
            activitySum = activitySum + 1
    if sp = learning then
        if activitySum < desiredActivity then
            active(c) = true
    else
        if activitySum < outputActivity then
            active(c) = true
```

III. Stable Sparse Coding

Our focus in this work was to compare the behaviour of ASP, nupicSP and rnASP, in terms of their ability to reach an equilibrium and produce stable encodings of auditory input, yet remain able to adapt to significantchanges in the input. We chose to use the birdsong dataset from the ICML 2013 Bird Challenge[2], which comprises song recordings of 35 species of birds. The recordings are 150 seconds in duration, sampled at 44.1 kHz and 16 bits, and contain ambient noise for which no attempt to remove was undertaken. Additionally, no technique was used to identify the onset and conclusion of the birdsongs within the files, as we consider this ability would be performed by a temporally-focussed neocortical function.

A. Preprocessing

The first stage of preprocessing involved converting the raw sound files into a data format that would preserve salient qualities of the bioacoustics. For this task we used a gammatone wavelet transform as it has been shown to be a well-performing model of the mammalian cochlea's basilar membrane dynamics [27]. By following the procedure described in [28], we generated wavelet outputs of 100 coefficients as our starting point for this work.

Our next consideration was the limitation imposed by nupicSP's inability to process real-valued input, i.e. it accepts *only* binary-valued input. As part of the supplementary tools provided with nupic, a range of scalar encoders are available to perform data conversion to binary format. Accordingly, we used the Scalar Encoder (SE) and Random Distributed Scalar Encoder (RDSE) which transform real-valued input into a bit mapped representation. Both scalar encoders output bit arrays that preserve the ordering of real-values by varying the bit patterns by at most one 'on' bit. While the SE encodes the real-value as a contiguous block of 1s set within the larger bit array of 0s, the RDSE randomly distributes the 1s throughout the bit array. We set the size of the output arrays to 100 bits. Consequently, for each of the 100 real-values of the wavelet transforms, we obtain an output of 100 bits, which we concatenated to form a 100×100 pixel matrix, where the each column of pixels is the scalar encoded real-value (see Figure 2). While this preprocessing step is counter-intuitive and biologically implausible in this context, it was necessary to allow comparison between ASP and nupicSP.

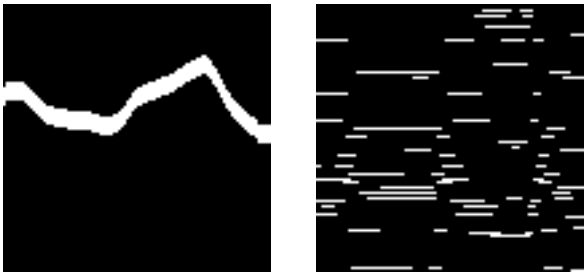

Fig. 2: Left: Scalar Encoder (SE). Right: Random Distributed Scalar Encoder (RDSE). Binary input data with a resolution of 100×100 pixels where each column of pixels is a single real-value from the gammatone wavelet transforms encoded using nupic binary encoders.

B. Learning Sparse Codes

The need to perform data conversion to binary format raised questions regarding the saliency of the transformed data. In order to assess the degree to which the altered data may have been impacted, we chose to perform classification tests on the column codes from ASP, nupicSP and rnASP.

The ICML 2013 Bird Challenge dataset, having been created for use in an open competition, does not provide ground-truths for the test set. To overcome this restriction, we used only the training set and employed 10-fold cross validation during classification. We applied an SVM classifier as the final step in the processing pipeline and manually tuned the SVM *C* parameter (i.e. the penalty factor which must be set to avoid over or under-fitting).

1) Stability of nupicSP: The three alternative variations of SP were repeatedly applied to both the SE and RDSE transformed data using 10 different random seeds. During this process, we monitored the dynamic behaviour of the systems by capturing metrics of changes (e.g. average number of connected synapses across all columns) occurring during each cycle through the dataset. While both ASP and rnASP were able to attain stable equilibriums where no further changes occurred (i.e. converged to solutions), nupicSP did not. In the absence of clear convergence of the nupic system, the most likely indication of movement towards a stable state was demonstrated by the average number of connected synapses

[2]Kaggle. https://www.kaggle.com/c/the-icml-2013-bird-challenge

at the end of each cycle steadily declining until it reach a stable value. Accordingly, we implemented a method which used this information to identify what we deemed as 'pseudo-convergence' and stopped the learning process.

For all three systems, the final output was a set of 100 values, where each value represented the activation level of the learned columns when re-encountering the input instance. It should be noted however, as the scalar encoders increased the input resolution of the data (from 100 real-values to 10,000 bits), the learning process in this context is performing dimension reduction, in addition to learning a sparse distributed representation.

We produced two variations of output: the first being composed of real-values for active columns calculated as the column $overlap \times boost$; and the second variation composed of binary activations, where 'on' is emitted from coding (i.e. active) columns and 'off' from non-coding columns. We report the results, summarised for both output variations from both SPs using SE and RDSE transformed data, as the average classification accuracy obtained by the SVM classifier as averaged across the ten SP seeds, and across ten SVM random seeds. (Refer Table I.)

	Overlap x Boost		**Binary**	
	SE Data	RDSE Data	SE Data	RDSE Data
nupicSE	68.4 %	66.8 %	64.1 %	62.1 %
ASP	72.6 %	72.3 %	70.3 %	70.2 %
rnASP	**75.8 %**	73.3 %	73.2 %	71.9 %

TABLE I: nupicSP, ASP and rnASP SVM classification accuracies using SE and RDSE binary transformed data. SDR output produced in two formats: column *overlap* × *boost* and binary column codes.

2) ASP vs rnASP: To complete the investigation regarding the retained saliency of the binary transformed data, we performed classification on the real-valued wavelet output using ASP and rnASP using the same methodology described above. These tests also allowed us to determine if the addition of different activity level parameters implemented in rnASP provided improved learning which would lead to better classification accuracies on audio input. Firstly, using the real-valued input and the previously identified *desiredActivity* level for ASP (that produced SDRs at an average of 10% sparsity), we reduced that value in rnASP by 25%, 50% and 75% and increased the corresponding *outputActivity* level to ensure an average output sparsity of 10% from rnASP. We executed these tests using ten random seeds and then performed SVM classification as described earlier. The averaged results are listed in Table II.

Output	**Overlap x Boost**	**Binary**
ASP	80.0 %	79.4 %
rnASP		
-75% *desiredActivity*	73.6 %	74.6 %
-50% "	**80.6 %**	80.3 %
-25% "	80.1 %	80.4 %

TABLE II: Classification accuracies obtained on real-valued data. SDR output produced in two formats: column *overlap* × *boost* and binary.

C. Stability of ASP and rnASP

Our initial and primary goal in this work was to assess the relative stability and adaptiveness of the SP algorithm. However, following the results of the previous experiments and due ot ASP's ability to process real-valued input (which is a more biologically plausible input format for auditory data), and nupicSP's inability to converge on a stable state (without learning being artificially halted), we focussed our attention on observing the behaviour of ASP and rnASP during the learning process.

These experiments were started by first training both systems until they reached a stable equilibrium (i.e. converged to a point attractor). We then systematically added random noise to the data to represent varying degrees of unpredictable changes in the environment to determine whether the systems began to learn again in this context. If, in the event that the systems subsequently did find a second stable state in the noisy environment, we reintroduced the uncorrupted data to investigate whether the second state had resulted in a more or less strongly adapted system.

For the real-valued data (previously normalised to values between 0 and 1), noise was introduced by randomly selecting a percentage of the 100 coefficients which were then adjusted by a randomly selected value in the range ± 0 to 0.5 (and ensuring the new values did not exceed the normalisation limits). We introduced noise in 10% increments up to a maximum of 100% and repeatedly executed the experiments using 20 different random seeds.

In the case of the binary encoded data, we elected to use the RDSE encoded set, and introduced noise to the data by randomly flip bits in the binary matrix. Noise was introduced in increments of 5% up to a maximum of 30%, and the experiments were again conducted using 20 different random seeds.

To capture the systems' responses to the noise, we recorded the average number of changed synapses over all columns during each cycle of learning, and as averaged over the set of random seed executions. Because the dimensions of the real-valued and binary-valued data are disproportionate, the available number of synapses is vastly different. Consequently, we report the average changed synapses as the percentage of total available synapses for each data format. Figures 3, 4 and 5graph the stability results from ASP, 6, 7 and 8 report results from rnASP.

IV. Discussion

The intention in this study was to focus on investigating the stability and adaptiveness of three variations of HTM's Spatial Pooler. However, due to nupicSP's inability to process real-valued input, we elected to also undertake a small series of classification tests to assess the saliency of the input data

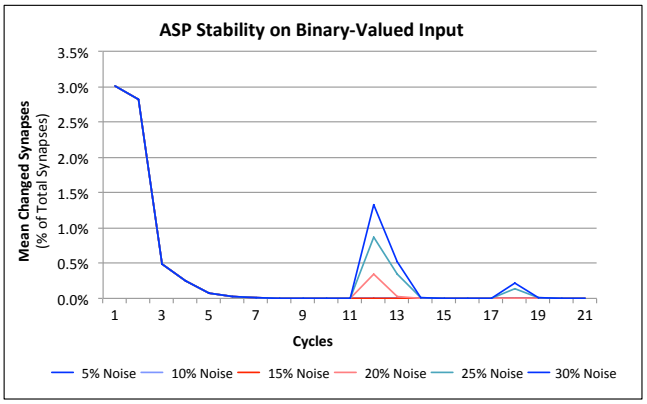

Fig. 3: ASP stability test on binary-valued input.

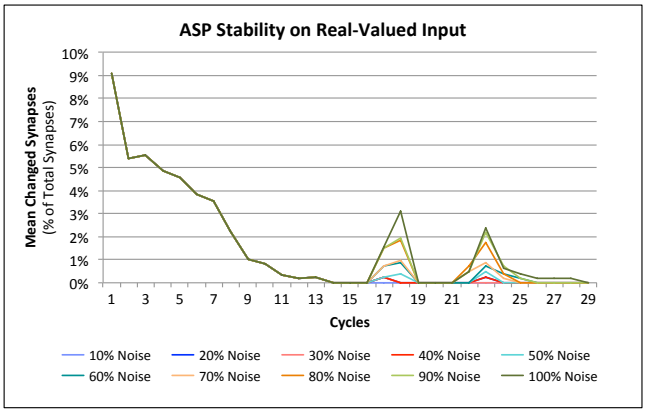

Fig. 4: ASP stability test on real-valued input.

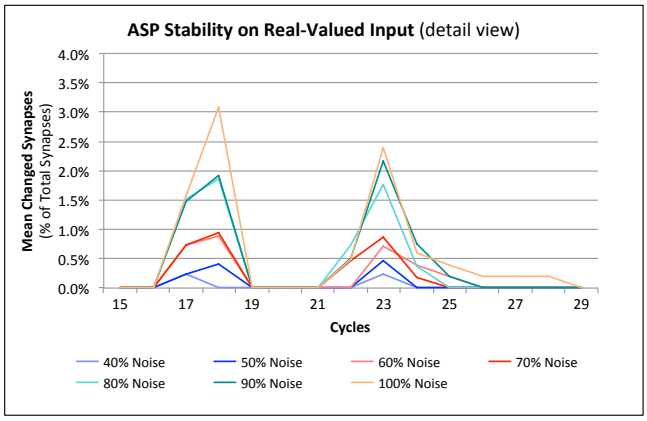

Fig. 5: ASP stability test on real-valued input, detailed view.

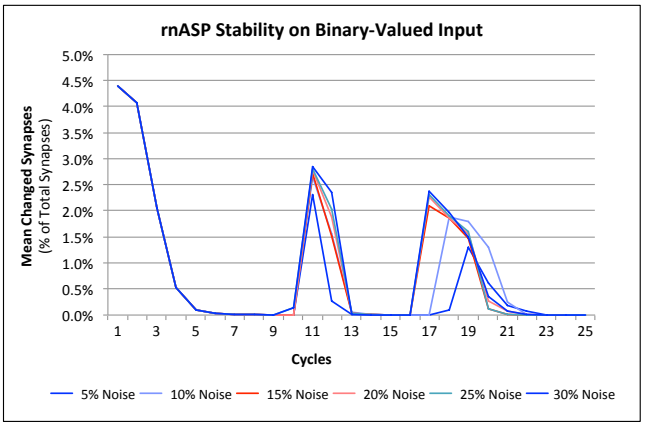

Fig. 6: rnASP stability test on binary-valued input.

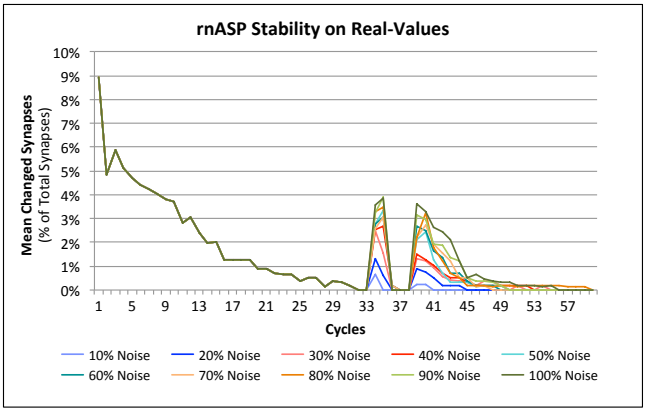

Fig. 7: rnASP stability test on real-valued input.

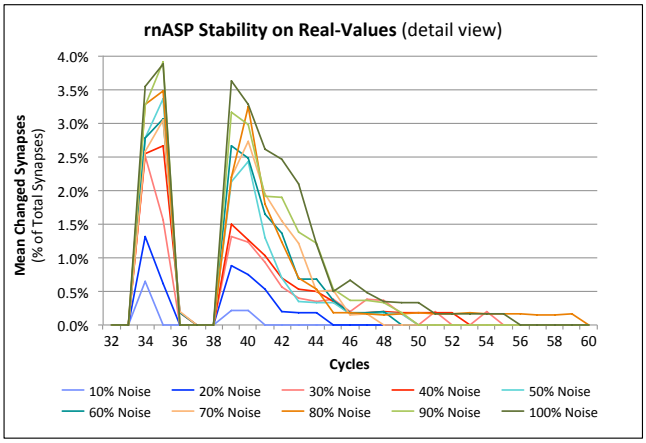

Fig. 8: rnASP stability test on real-valued input, detailed view.

once it had been transformed into suitable binary format for nupicSP. The results of these tests showed that recoding the birdsong wavelet data into a binary format reduced the classification accuracy that may be obtained as compared to results obtained using real-valued input.

The comparison undertaken using the binary encoded version of the birdsong data, where two different types of output encodings were produced (refer Table I), showed that higher classification accuracies obtained when the output was real-valued (i.e. column *overlap X boost*) compared to output that was composed of simple binary patterns of the active columns. We believe this is due to the learned synaptic patterns capturing subtle distinguishing characteristics of the input which are preserved by the boosting factor during output. In contrast, the binary output effectively discards that fine detail, providing only a generalised encoding of the data. The best classification result of 75.8% was achieved by ASP on

the SE binary encoding, and generally ASP performed more consistently compared to both rnASP and nupicSP. These result suggests that ASP is more responsive to subtle changes in the input structure.

The main failing of nupicSP, however, is its inability to converge to a stable equilibrium state without some form of external intervention that switches off its learning capabilities. This means it is unable to respond to subsequent changes in input without having its learning capabilities switched on again. While such approach is feasible for controlled experiments on benchmark problems, it does not represent the kind of sensory encoding required for a genuine adaptive system. In contrast, ASP is able to converge on equilibrium states on all the data sets considered here, and is further able to reinitiate learning without supervision when the input is sufficiently perturbed.

The use of an additional activity parameter in rnASP appears to afford the system better learning in comparison to ASP. The results of applying rnASP to the binary encoded data are particularly interesting as the same trend in classification rates across the range of SE encoder sizes is closely matched for both ASP and rnASP, with the main difference being that rnASP performs better for both output types. We consider this a strong indication that the restricted neighbourhood approach does successfully enforce more learning without altering the more general behaviour of the system. However, the difference in classification accuracy on the real-valued data is less significant, with rnASP only managing to achieve a minor improvement in the *overlap X boost* output codes. This is most likely due to ASP already being able to maximise the information contained within the real-valued input, and therefore leaves little remaining information to be extracted by more intensive learning. In contrast, the use of the more constrained neighbourhood does improve the classification accuracy that may be obtained from the binary encoded output and again confirms the technique is a valuable one when applied in this context.

Initial investigation into the adaptivity of ASP and rnASP using the binary encoded data showed that both systems become well adapted to this form of input. Both ASP and rnASP remained stable for noise percentages up to 15%, and above that level both systems were able to adapt to the changed input. The significant difference between the two systems on noise levels from 20% is that rnASP responds more strongy with a higher percentage of synapses changing, and requires significantly more cycles to attain stability. This would be a direct consequence of the restricted neighbourhood forcing more columns to undergo further synaptic learning.

A similar trend is see again when using real-valued input, where rnASP responds with greater numbers of changing synapses. However, two points of difference are seen: firstly, ASP is still stable at low levels of noise as compared to rnASP which begins to adapt from the initial introduction of noise at 10%. Secondly, both systems demonstrate noticeable phase transitions, occuring at approximate noise level increases of 20%. Despite these distinctive transitions in the number of synapses changing, both systems are still able to adapt to the arrival of changed input. Lastly, for both systems, neither has become fully adapted to the corrupted information, such that upon reintroduction of the original, uncorrupted data, they are both able to again commence learning and re-adapt, and in line with the generally observed behaviour of rnASP, its recovery requires more synapses to adapt and the system again takes significantly more cycles to reach the new stable state.

V. Conclusion

In this study we compared two alternative versions of the HTM Spatial Pooler, ASP and nupicSP, in terms of their ability to produce SDRs of birdsongs. The results showed that ASP was able to better capture significant features of the input for the purposes of classification. To the best of our knowledge, this is the first evaluation of nupicSP when applied in the domain of auditory processing.

The failure of nupicSP to display clear convergence behaviour casts some doubt on the degree to which its current implementation adequately models pertinent aspects of the neocortex for sensory encoding. ASP's demonstrated ability to self-organise convergence allows us to consider it a more biologically more plausible model of neocortical function.

As an extension to ASP, we introduced an enhanced approach to the inhibition processes in the rnASP algorithm that increases the learning capacity of the system, without undue impact on its ability to self-organise and find stable attractors on binary data.

Additional contributions of this work include an initial evaluation of ASP from an adaptive systems perspective, where we identified the conditions under which the system remains stable to environmental perturbation. We further evaluated the relative adaptivity and stability of both ASP and rnASP on binary and real-valued data, and determined that their behaviour tends to remain adaptive as more noise is introduced.

As ASP (and by extension rnASP) are only the first step of the process performed within a region of the HTM model, which should function within a hierarchy with contextual feedback from higher levels, we consider the model worthy of further investigation. We look towards incorporating the new rnASP into the larger HTM model to further investigate its behaviour in that wider context. Further work would also include investigating the TP from the adaptive systems perspective.

References

[1] J. Hohwy, *The predictive mind*. Oxford University Press, 2013.

[2] A. Clark, *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press, 2015.

[3] R. Kanai, Y. Komura, S. Shipp, and K. Friston, "Cerebral hierarchies: predictive processing, precision and the pulvinar," *Phil. Trans. R. Soc. B*, vol. 370, no. 1668, p. 20140169, 2015.

[4] E. T. Rolls, "Cortical coding," *Language, Cognition and Neuroscience*, vol. 32, no. 3, pp. 316–329, 2017.

[5] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[6] D. J. Field, "What is the goal of sensory coding?" *Neural computation*, vol. 6, no. 4, pp. 559–601, 1994.

[7] B.-H. Juang and L. R. Rabiner, "Automatic speech recognition–a brief history of the technology development," 2004, http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf (Last accessed March 31, 2017).

[8] D. Stowell and M. D. Plumbley, "Birdsong and c4dm: A survey of uk birdsong and machine recognition for music researchers," *Centre for Digital Music, Queen Mary University of London, Tech. Rep. C4DM-TR-09-12*, 2010.

[9] B. Goertzel, R. Lian, I. Arel, H. De Garis, and S. Chen, "A world survey of artificial brain projects, part ii: Biologically inspired cognitive architectures," *Neurocomputing*, vol. 74, no. 1, pp. 30–49, 2010.

[10] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including htm cortical learning algorithms," Tech. rep., Numenta, Inc, Palto Alto (2006), 2011, https://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf (Last accessed February 27, 2017).

[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[12] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, 2009, pp. 1601–1608.

[13] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," *arXiv preprint arXiv:1511.06309*, 2015.

[14] K. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[15] J. Hawkins and S. Blakeslee, *On intelligence*. Macmillan, 2007.

[16] M. Awad and R. Khanna, "Deep learning," in *Efficient Learning Machines*. Springer, 2015, pp. 167–184.

[17] M. Wielgosz and M. Pietroń, "Using spatial pooler of hierarchical temporal memory to classify noisy videos with predefined complexity," *Neurocomputing*, vol. 240, pp. 84–97, 2017.

[18] J. Balasubramaniam, C. G. Krishnaa, and F. Zhu, "Enhancement of classifiers in htm-cla using similarity evaluation methods," *Procedia Computer Science*, vol. 60, pp. 1516–1523, 2015.

[19] J. Mnatzaganian, E. Fokoué, and D. Kudithipudi, "A mathematical formalization of hierarchical temporal memory's spatial pooler," *Frontiers in Robotics and AI*, vol. 3, p. 81, 2016.

[20] M. Pietro, M. Wielgosz, and K. Wiatr, "Formal analysis of htm spatial pooler performance under predefined operation conditions," in *Rough Sets: International Joint Conference, IJCRS 2016, Santiago de Chile, Chile, October 7–11, 2016, Proceedings*, vol. 9920. Springer, 2016, pp. 396–405.

[21] M. Leake, L. Xia, K. Rocki, and W. Imaino, "A probabilistic view of the spatial pooler in hierarchical temporal memory," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 5, pp. 1266–1273, 2015.

[22] A. Kneller and J. Thornton, "Distal dendrite feedback in hierarchical temporal memory," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[23] A.-J. Perea-Moreno, A. García-Cruz, N. Novas, and F. Manzano-Agugliaro, "Rooftop analysis for solar flat plate collector assessment to achieving sustainability energy," *Journal of Cleaner Production*, vol. 148, pp. 545–554, 2017.

[24] A. K. Sungur and E. Surer, "Voluntary behavior on cortical learning algorithm based agents," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–7.

[25] J. Thornton and A. Srbic, "Spatial pooling for greyscale images," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 3, pp. 207–216, 2013.

[26] L. Main, B. Cowley, A. Kneller, and J. Thornton, "Evaluating sparse codes on handwritten digits." in *Australasian Conference on Artificial Intelligence*. Springer, 2013, pp. 396–407.

[27] A. Venkitaraman, A. Adiga, and C. S. Seelamantula, "Auditory-motivated gammatone wavelet transform," *Signal Processing*, vol. 94, pp. 608–619, 2014.

[28] L. Main and J. Thornton, "A cortically-inspired model for bioacoustics recognition," in *International Conference on Neural Information Processing*. Springer, 2015, pp. 348–355.