Coupled Neural Associative Memories

Amin Karbasi[1], Amir Hesam Salavati[2] and Amin Shokrollahi [3]

***Abstract*— We propose a novel architecture to design a neural associative memory which is capable of learning a large number of patterns and recalling them later in presence of noise. Based on recent developments in spatially-coupled codes, we show that the performance of our proposed network in eliminating noise is drastically better than the previous approaches while it also maintains the ability of learning an exponentially large number of patterns. Previous work either failed in providing good performance during the recall phase or in offering large pattern retrieval (storage) capacities.**

The proposed architecture is based on dividing the neurons into local clusters and parallel plains, very much similar to the architecture of the visual cortex of macaque brain [17]. The common features of our proposed architecture with the spatially-coupled codes on graphs enable us to benefit from the analytical approaches developed in coding theory to assess the performance of the neural algorithms in the recall phase of the associative memories.

The main contribution of this paper is to show how a neural network based on simple building blocks with limited capabilities can in fact achieve significant performances in correcting input errors and retrieve the correct answer among a huge number of previously memorized patterns.

I. Introduction

The ability of the brain to memorize large quantities of data and later recalling them from partially available information is truly staggering. While relying on iterative operations of simple (and sometimes faulty) neurons, our brain is capable of retrieving the correct "memory" with high degrees of reliability even when the cues are limited or inaccurate to a large extent.

Not surprisingly, designing artificial neural networks capable of accomplishing this task, called *associative memory*, has been a major point of interest in the neuroscience community for the past three decades. Starting with the seminal work of Hopfield [1], there have been lots of artificial neural associative memories capable of memorizing a given data set and reliably retrieving memorized patterns later in response to noisy queries.

This problem, in its core, is in fact very similar to the issue of reliable information transmission faced in communication systems where the goal is to find mechanisms to efficiently encode and decode a set of transmitted patterns over a noisy channel.

[1]Department of Computer and Communication Sciences,Ecole Polyetechnique Federale de Lausanne, Lasuanne, Switzerland amin.karbasi@epfl.ch

[2]Department of Computer and Communication Sciences,Ecole Polyetechnique Federale de Lausanne, Lasuanne, Switzerland hesam.salavati@epfl.ch

[3]Department of Computer and Communication Sciences,Ecole Polyetechnique Federale de Lausanne, Lasuanne, Switzerland amin.shokrollahi@epfl.ch

More interestingly, the novel techniques employed to design good codes are extremely similar to those used in designing and analyzing neural networks. In both cases, graphical models, iterative algorithms and message passing play a central role.

However, despite the similarity in the task and the techniques, we witness a huge gap in terms of the efficiency achieved by them. More specifically, by using modern coding techniques, we are capable of reliably transmitting 2^{rn} binary vectors of length n over a noisy channel, where $0 < r < 1$. This is achieved by intelligently introducing redundancy among the transmitted messages, which will be later used to recover the correct pattern from the received noisy version. On the other hand, until recently, artificial neural associative memories were only capable of memorizing $O(n)$ binary patterns of length n (see, for example, [1], [2], [3], [4]).

Part of the reasons for this gap goes back to the assumption held in the mainstream work on artificial associative memories, which requires the network to be able to memorize *any* set of randomly chosen binary patterns. While it gives the network a certain degree of versatility, it severely hinders the efficiency.

To achieve an exponential scaling in the storage capacity of neural networks Kumar et al. [5] suggested a different viewpoint in which the network is no longer required to memorize *any* set of random patterns but only those that have some *structure* within them. In their paper, this structure has been captured by ensuring that patterns all belong to a subspace with dimension $k < n$. By assuming that the connectivity matrix of the neural graph is given, the authors proposed a simple iterative algorithm for the recall phase. However, they did not propose any algorithm for the learning phase of the association problem. This task was then accomplished in [11] and [16].

Although the proposed approaches are capable of achieving exponential scaling for the pattern retrieval capacity, the performance of the algorithms employed in the recall phase are still far from being desirable.

In this work, we follow the same viewpoint as in [5] where we only focus on memorizing a set of patterns with some degree of redundancy. However, we propose a different neural architecture to capture local correlations among patterns, similar to the way that the receptive field in human visual cortex is arranged. We use the algorithm proposed in [16] for the learning phase. For the recall phase, we also employ the algorithm proposed in [11] as a building block. However, due to the novel structure, we will need a more powerful tool to investigate the performance of the recall algorithm. This is done by applying recent developments in the field of

spatially-coupled codes [14], [13].

II. Related Work

The first artificial neural auto-associative memory was introduced in the pioneering work of Hopfield [1]. The "Hopfield network" is based on a complete graph of n neurons which utilize the Hebbian learning rule [6] to memorize a set of *randomly* chosen binary patterns of length n. Later on, McEliece et al. derived the pattern retrieval capacity of Hopfield networks and showed that it is equal to $\mathcal{C} = (n/2\log(n))$ [7].

Since then, there have been many attempts to increase the pattern retrieval capacity of neural associative memories while keeping them as versatile as possible, i.e. maintaining the ability to memorize any set of randomly chosen patterns. Venkatesh et al. [2] sacrificed the online learning capability to increase the capacity by calculating the weight matrix offline using the pseudo-inverse rule [12]. The result is a pattern retrieval capacity of $n/2$ random patterns with the ability of one bit error correction. Jankowski et al. [3] investigated a different model with multi-state associative memory in which each neuron can be assigned a multivalued state from the set of complex numbers. It was shown by Muezzinoglu et al. [4] that the capacity of such networks can be increased to $\mathcal{C} = n$ at the cost of a prohibitive weight computation mechanism.

More recently, a new perspective has been put to use the goal of which is to design neural associative memories that memorize only patterns with some degree of inherent redundancy. In this framework, a tradeoff is being made between versatility, i.e. the capability of the network to memorize any set of random patterns, and the pattern retrieval capacity. Pioneering this frontier, Berrou and Grippon considered memorizing patterns based on Walsh-Hadamard codes, frequently used in CDMA communications for their low-correlation properties [9]. While the suggested approach increases the capacity beyond n, the complexity of operations in the recall phase was prohibitive.

Gripon and Berrou [8] recently proposed another method based on neural clicks which increases the pattern retrieval capacity potentially to $O(n^2/\log(n))$ with a low complexity algorithm in the recall phase. The proposed approach is based on memorizing a set of patterns mapped from randomly chosen binary vectors of length $k = O(\log(n))$ to the n-dimensional space. This way, one could benefit from the extra redundancy within the patterns to increase the capacity as well as correcting errors, much in the same way error correcting codes do in communication systems.

Further increases in the pattern retrieval capacities can be achieved by considering neural models based on higher order correlations, as opposed to pairwise correlation considered in Hopfield networks. Peretto et al. [10] showed that the pattern retrieval capacity of such networks can be improved to $\mathcal{C} = O(n^{p-2})$, where p is the degree of correlation. The main drawback of this and similar methods lies in the prohibitive computational complexity required by the learning phase.

By considering patterns that come from a subspace with dimension $k < n$, Kumar et al. were able to show an exponential scaling in the pattern retrieval capacity, i.e. $\mathcal{C} = O(a^n)$, with some $a > 1$. This model was later extended to modular patterns, i.e. those in which the pattern is divided into non-overlapping sub-patterns where each sub-pattern come from a subspace [11]. The authors have proposed a simple iterative learning algorithm as well as achieving an improved performance in the recall phase as compared to [5].

In this paper, we follow the same line of work by extending the model proposed in [11] to neural networks with modular structure in three dimensions, similar to the way visual cortex of the macaque brain is organized [17]. The proposed model is based on *overlapping* local clusters of a number of neighboring neurons, where clusters are arranged in parallel planes. However, there are sparse connections between various clusters in different planes as well. We then focus on memorizing those patterns with the property that local sub-patterns in the domain of each cluster show a certain degree of correlation and redundancy.

Interestingly, this model is very similar to spatially-coupled codes on graph [13]. Therefore, we will borrow analytical tools developed to analyze such codes [14] in order to investigate the performance of the proposed error correcting algorithm during the recall phase.

It should also be mentioned that the suggested model is particularly similar to the spatially-coupled Generalized LDPC code (GLDPC) with Hard Decision Decoding (HDD) proposed in [15]. In our model, the clusters can be interpreted as "component codes" in the GLDPC ensemble and the parallel planes as spatial-coupling of the individual decoders. However, in contrast to [15], the proposed error correction algorithm in this paper is purely based on simple operations and message passing even within clusters (component codes). This makes it specially suitable for the main purpose of this paper which is to develop an associative memory suitable for neural networks with their inherent limitations on the type and complexity of operations. Furthermore, because of limitations in the neural networks, a message transmitted by a neuron goes to all its neighbors since it can not single out one of its neighbors to send a different message. This assumption translates into the fact that each iteration of the algorithm corresponds to a matrix-vector multiplications, which significantly reduces the complexity of the algorithm.

III. Model and Notations

The main goal of this paper is to design a neural associative memory capable of learning a large number of patterns while being able to eliminate a large amount of noise during the recall phase. More specifically, the target is to memorize $\mathcal{C}$ patterns of length n from a data set $\mathcal{X}$ (learning phase), and being able to return the correct memorized pattern in response to noisy queries (recall phase). Furthermore, we would like to achieve a large capacity, preferably, $\mathcal{C} = O(e^n)$.

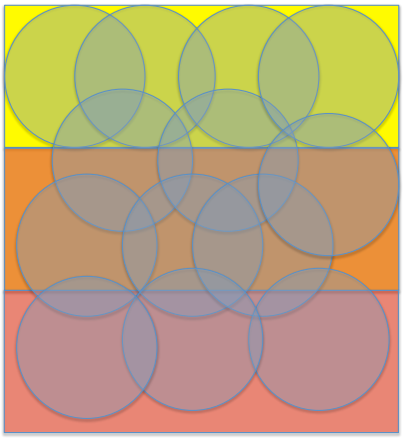


Fig. 1: A rough sketch of the proposed model illustrated as the receptive fields for processing visual signals. Each circle represents a cluster. The planes are specified by rectangles of different colors.

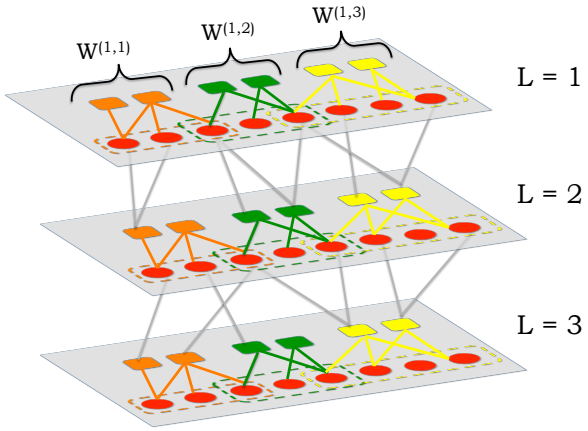


Fig. 2: The proposed architecture of a coupled neural associative memory

We assume that the patterns are integer-valued, with each entry $x_i \in \mathcal{S} = \{0, \ldots, S-1\}$.[1] We also focus on only memorizing patterns with strong *local correlation* among their entries. This correlation is captured in the form of negligible minor components among sub-patterns of the dataset.

More specifically, we divide each pattern into L divisions of the same length[2], which we call *planes*. Within each plane, we further divide the patterns into D *overlapping* clusters. Here, overlapping refers to the fact that an entry in a pattern can lie within the domain of multiple clusters within the same plane or other planes. We assume that each pattern neuron in plane ℓ is connected to at least one cluster in planes $\ell-\Omega, \ldots, \ell+\Omega$ (except at the boundaries). Therefore, each pattern neuron is connected to $2\Omega+1$ planes, on average.

Furthermore, we consider memorizing those patterns in which the entries of each pattern in each cluster are correlated to each other in the sense that they belong to a subspace or have negligible minor components.

A better way of understanding the model is to consider its application on $2D$ data such as images (see Figure 1). In this example, each row of the image corresponds to the planes. Clusters on the other hand are the overlapping *receptive fields*, which cover an area composed of neighboring neurons in different planes. This is in fact very similar to the configuration of the receptive fields in human visual cortex [18]. Our assumptions on strong correlations then translates into assuming strong dependencies within each receptive field for all patterns in the dataset.

Learning phase: This phase corresponds to learning the minor components or dual vectors for each cluster. In this paper, we use the learning algorithm proposed in [16]. The output of the learning algorithm is a matrix $W^{(\ell,d)}$ for the cluster d in plane ℓ. The rows of this matrix correspond to the dual vectors and the columns correspond to the pattern nodes. Therefore, letting $\mathbf{x}^{(\ell,d)}$ denote the sub-pattern corresponding to the domain of cluster d of plane ℓ, we will have

$$W^{(\ell,d)} \cdot \mathbf{x}^{(\ell,d)} = \mathbf{0}, \tag{1}$$

By treating these matrices as connectivity matrices of the neural graph, one can consider each cluster as a bipartite graph composed of *pattern* and *constraint* neurons. The constraint neurons do not have any overlap (i.e. each one belongs only to one cluster) whereas the pattern neurons can have connections to multiple clusters. We aim to keep these connections low to ensure good error correction capabilities later.

Putting all the local connectivity matrices together, we obtain the model shown in Figure 2 which shows the planes, the clusters, and the overall structure of the network. Interestingly, this model is very similar to the neural architecture to process visual signals in macaque brain [17].

Recall phase: The second major task of a neural associative memory is to retrieve correct memorized patterns in response to noisy queries. Here, the neural graph is learned (fixed) and we are looking for a simple iterative algorithm to eliminate noise from the query. In this paper, we assume that the the noise is additive, i.e., the query is one of the memorized patterns plus some noise. The noise itself is integer-valued and for simplicity we assume that its entries are $\{-1, 0, +1\}$, where a -1 $(+1)$ noise corresponds to a neuron skipping a spike (fire one more spike) than expected. More specifically, if the initial error probability is denoted by p_e, then each entry of the noise vector is $+1$ or -1 with probability $p_e/2$. In this paper, we propose an algorithm to eliminate this input noise.

Pattern Retrieval Capacity: Finally, an ideal neural associative memory should have a large pattern retrieval capacity. This is the maximum number of patterns that can

[1] A natural way of interpreting this assumption is to consider the entries as the short-term firing rate of corresponding neurons.

[2] The assumption on having the same lengths is only for simplicity of notations and can be generalized to variable lengths in a straightforward manner.

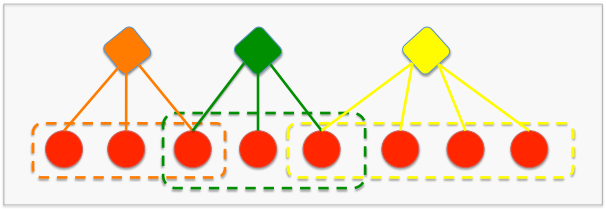


Fig. 3: The connectivity graph with neural planes and "super nodes" (shown in diamond-sahped nodes). Super nodes are composed of all constraints of a cluster. Here, a given example corresponding to the plane 1 of Figure 2 is shown.

be memorized by a network while still being able to return reliable responses in the recall phase. Here, as in [11], we look for exponentially large capacities, i.e. those for which $\mathcal{C} = O(a^n)$, with some $a > 1$.

A. Notations

In this paper, bold face letters represent vectors, captial letters represent matrices[3], subscripts represent entries in vectors or matrices, and superscripts index the clusters within planes and the network. As an example, $W^{(\ell,d)}$ represents the the connectivity matrix of the cluster d in plane ℓ and $W_{ij}^{(\ell,d)}$ is the (i,j) entry in this matrix.

In our model, we have L planes with n/L pattern neurons. Each plane contains D clusters. Cluster d in plane ℓ contains $m_{\ell,d}$ constraints neurons and is connected to $n_{\ell,d}$ pattern neurons. Note that due to overlaps among clusters, we have $\sum_d n_{\ell,d} > n/L$.

We also consider the overall connectivity graph of the plane ℓ, denoted by $\tilde{W}^{(\ell)}$, in which the constraint nodes in each cluster are compressed into one *super node*. Any pattern node that is connected to a given cluster is connected with an (unwedighted) edge to the corresponding super node. Figure 3 illustrate this graph for different planes.

For the graph $\tilde{W}^{(\ell)}$ we define the edge degree distribution as follows: let $\lambda_i^{(\ell)}$ and $\rho_j^{(\ell)}$ be the fraction of edges connected to pattern and constraint nodes with degree i and j, respectively, with $\sum_i \lambda_i^{(\ell)} = 1$ and $\sum_j \rho_j^{(\ell)} = 1$, for all planes ℓ. Finally, we define the degree distribution polynomials $\lambda^{(\ell)}(x) = \sum_i \lambda_i^{(\ell)} x^i$ and $\rho^{(\ell)}(x) = \sum_j \rho_i^{(\ell)} x^j$. We call $\lambda^{(\ell)}$ and $\rho^{(\ell)}$ the pattern and constraint degree distributions from an *edge perspective* in plane ℓ.

IV. Recall Phase

A. The Proposed Algorithm

The proposed recall algorithm in this paper is the extension of the one we proposed in [11] to coupled neural networks. For the sake of completeness, we have summarized this approach in Algorithm 1. We apply Algorithm 1 [11] *sequentially* to the neural planes in Figure 2, i.e. first we eliminate noise in the first plane, then the second one and so on. Once we have iterated over all planes, we go back and repeat this process as long as necessary.

Within each plane, we apply a sequential scheduling again. More specifically, we apply Algorithm 1 to the first neural cluster. Once finished, we check if all errors *within that cluster* have been corrected (simply by checking the final state of the constraint nodes, expecting to be all zero). If not, we revert the values of the pattern nodes in that cluster to their intial value (before applying error correction), otherwise keeping the current seemingly correct values. We then proceed to the next cluster until all clusters within the neural plane are covered. We repeat this process by going back to the first cluster as many times as necessary. Repeating this process will be helpful since some of the errors might have been corrected in the previous round, resulting in clusters that had two or more errors in their domain ending up with only a single error in this iteration. Therefore, they can correct this error now, helping their neighboring clusters. Algorithm 2 summarizes the proposed algorithm.

Algorithm 2 Error Correction of the Coupled Network

Input: Connectivity matrix of all clusters ($W^{(\ell,d)}, \forall\ell, \forall d$), iteration $t_{\max}$
Output: Correct memorized pattern $\mathbf{x} = [x_1, x_2, \ldots, x_n]$

```
for t = 1 → t_max do
  for ℓ = 1 → L do
    for d = 1 → D_ℓ do
      Apply Algorithm 1 to the cluster d of neural plane ℓ.
      Update the value of patterns nodes x^(ℓ,d) only if all the constraints in the clustered are satisfied.
      i ← i + 1
    end for
    ℓ ← ℓ + 1[a]
  end for
  t ← t + 1
end for
```

[a]Note that we can stay in the loop several time for each plane and move to the next plane only after we are fully done with the current one.

We consider two variants of the above error correction algorithm. In the first one, which we call the *constrained coupled neural error correction*, we provide the network with some side information during the recall phase. This is equivalent to "freezing" a few of the pattern neurons in the state known to be correct. This is similar to the spatially-coupled error correcting codes and is shown to achieve near optimum performances in many cases [13], [14]. In the case of neural associative memory, the side information can come from the context. For instance when trying to fill the blank in the sentence "The _ *at* flies", one can use the side information (flying) to decide the correct answer among multiple choices. Without this side iformation, we can not tell if _ *at* corresponds to *bat* or cat.

In the other variant, which is called *unconstrained coupled neural error correction*, we perform the error correction

[3]Except for L and D which represent the number of neural planes and clusters within each plane, respectively.

Algorithm 1 Error Correction Within Cluster [11]

Input: Connectivity matrix of the cluster ($W^{(\ell,d)}$), update threshold φ, iteration $t_{\max}$

Output: Correct memorized sub-pattern $\mathbf{x}^{(\ell,d)} = [x_1^{(\ell,d)}, \ldots, x_{n_{\ell,d}}^{(\ell,d)}]$

1: **for** $t = 1 \to t_{\max}$ **do**

2: *Forward iteration:* Calculate the weighted input sum $h_i = \sum_{j=1}^{n} W_{ij}^{(\ell,d)} x_j^{(\ell,d)}$, for each neuron $y_i^{(\ell,d)}$ and set:

$$y_i^{(\ell,d)} = \begin{cases} 1, & h_i < 0 \\ 0, & h_i = 0 \\ -1, & \text{otherwise} \end{cases}$$

3: *Backward iteration:* Each neuron $x_j^{(\ell,d)}$ computes

$$g_j^{(\ell,d)} = \frac{\sum_{i=1}^{m_{\ell,d}} W_{ij}^{(\ell,d)} y_i^{(\ell,d)}}{\sum_{i=1}^{m_{\ell,d}} |W_{ij}^{(\ell,d)}|}.$$

4: Update the state of each pattern neuron j according to $x_j^{(\ell,d)} = x_j^{(\ell,d)} + \text{sgn}(g_j^{(\ell,d)})$ only if $|g_j^{(\ell,d)}| > \varphi$.

5: $t \leftarrow t + 1$

6: **end for**

without providing any side information. This would be similar to many other neural approaches for the recall phase and will be used as a benchmark to compare the proposed algorithm with those of other work [5], [11].

B. Analysis of the Recall Algorithm

We start the analysis by considering a theorem in [11] stating that Algorithm 1 is capable of correcting at least one bit of initial error with high probability.

Theorem 1: [11] When $\varphi \to 1$, Algorithm 1 can correct a single error in each cluster with probability at least $1 - \left(\frac{\bar{d}}{m}\right)^{d_{min}}$, where $\bar{d}$ and d_{min} are the average and minimum degree of the pattern nodes within the cluster domain.

Knowing this fact, let us investigate the error correction algorithm within a single plane. Let $z^{(\ell)}(t)$ denote the *average* probability of error for pattern nodes across neural plane ℓ and in iteration t. Furthermore, let $\pi^{(\ell)}(t)$ be the *average* probability of a *custer* neuron in plane ℓ sending an erroneous message to its neighbors. We will derive recursive expressions for $z^{(\ell)}(t)$ and $\pi^{(\ell)}(t)$.

A cluster node in plane ℓ receives noisy messages from its neighbors with an average probability of $\bar{z}^{(\ell)}$, where

$$\bar{z}^{(\ell)} = \frac{1}{2\Omega+1} \sum_{j=-\Omega}^{\Omega} z^{(\ell-j)} \tag{2}$$

with $z^{(i)} = 0$ for $i \leq 0$ and $i > L$.

Let $\pi_i^{(\ell)}$ denote the the probability that a cluster node with degree i in plane ℓ sends an erroneous message to its neighboring pattern nodes. Then, knowing that each cluster is capable of correcting at least one error, $\pi_i^{(\ell)}$ is equal to the probability of receiving two or more noisy messages from pattern neurons,

$$\pi_i^{(\ell)} = 1 - \left(1 - \bar{z}^{(\ell)}\right)^i - i\bar{z}^{(\ell)} \left(1 - \bar{z}^{(\ell)}\right)^{i-1} \tag{3}$$

Now, letting $\pi^{(\ell)}(t)$ denote the average probability of sending erroneous nodes by cluster nodes in plane ℓ and in iteration t, we will have

$$\begin{aligned} \pi^{(\ell)}(t) &= \mathbb{E}\{\pi_i^{(\ell)}\} = \sum_i \rho_i \pi_i^{(\ell)} \\ &= 1 - \rho(1 - \bar{z}^{(\ell)}(t)) + \bar{z}^{(\ell)}(t)\rho'(1 - \bar{z}^{(\ell)}(t)) \end{aligned} \tag{4}$$

where $\rho(z) = \sum_i \rho_i z^i$ is the cluster node degree distribution polynomial and $\rho'(z) = d\rho(z)/dz$.

To simplify notations, let us define the function $g(z) = 1 - \rho(1 - z) + z\rho'(1 - z)$ such that

$$\pi^{(\ell)}(t) = g(\bar{z}^{(\ell)}(t)) \tag{5}$$

Now consider a given pattern neuron with degree j in plane ℓ. In iteration $t+1$, Let $z_j^{(\ell)}(t)$ denote the probability of sending an erroneous message by this node. Then, $z_j^{(\ell)}(t)$ is equal to the probability of this node being noisy in the first place (p_e) and having all its cluster nodes sending erroneous messages in iteration t, the average probability of which is $\bar{\pi}^{(\ell)}(t) = \frac{1}{2\Omega+1} \sum_{i=-\Omega}^{\Omega} \pi^{(\ell-i)}(t)$. Now, since $z^{(\ell)}(t+1) = \mathbb{E}\{z_j^{(\ell)}(t+1)\}$, we get

$$\begin{aligned} z^{(\ell)}(t+1) &= p_e \sum_j \lambda_j \left(\bar{\pi}^{(\ell)}\right)^j \\ &= p_e \lambda(\bar{\pi}^{(\ell)}) \\ &= p_e \lambda(\frac{1}{2\Omega+1} \sum_{i=-\Omega}^{\Omega} g(\bar{z}^{(\ell-i)}(t)) \end{aligned} \tag{6}$$

Again to simplify the notation, let us define the function $f(z; p_e) = p_e \lambda(z)$. This way, we will have the recursion as:

$$z^{(\ell)}(t+1) = f(\frac{1}{2\Omega+1} \sum_{i=-\Omega}^{\Omega} g(\bar{z}^{(\ell-i)}(t); p_e) \tag{7}$$

Therefore, the decoding operation will be successful if $z^{(\ell)}(t+1) < z^{(\ell)}(t)$, $\forall \ell$. As a result, we look for the maximum p_e such that we will have $f(\frac{1}{2\Omega+1} \sum_{i=-\Omega}^{\Omega} g(\bar{z}^{(\ell-i)}(t); p_e) < z^{(\ell)}$ for $z^{(\ell)} \in [0, p_e]$. Let p_e^* denote this probability for the constrained coupled system and $p_e^\dagger$ represent the same quantity for the unconstrained system.

We will use the analytical approaches recently proposed in [14] to derive this threshold. To this end, a potential function is defined ro the system which will be used to track the iterative evolution of the error probability in equation (6). Let $\mathbf{f}(z; p_e) : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g}(z) : \mathbb{R}^n \to \mathbb{R}^n$ be two component-wise vector functions operating on a vector $\mathbf{z}$ such that $[\mathbf{f}(\mathbf{z}; p_e)]_i = f(z_i; p_e)$ and $[\mathbf{g}(\mathbf{z})]_i = g(z_i)$, where $f(z_i; p_e)$ and $g(z_i)$ are defined earlier in recursions (5) and (7). Using this defintion, one can rewrite recursion (6) in the vector form as [14]:

$$\mathbf{z}(t+1) = A^\top \mathbf{f}(A\mathbf{g}(\mathbf{z}(t)); p_e) \tag{8}$$

where A is the *coupling matrix* defined as[4]:

$$A = \frac{1}{2\Omega+1}\begin{bmatrix} \overbrace{1 & 1 & \cdots & 1}^{\Omega} & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

At this point, the potential function of the unconstrained coupled system could be defined as [14]:

$$\begin{aligned} U(\mathbf{z}; p_e) &= \int_C \mathbf{g}'(\mathbf{u})(\mathbf{u} - A^\top \mathbf{f}(A\mathbf{g}(\mathbf{u})).d\mathbf{u} \\ &= \mathbf{g}(\mathbf{z})^\top \mathbf{z} - G(\mathbf{z}) - F(A\mathbf{g}(\mathbf{z}); p_e) \end{aligned} \quad (9)$$

where $\mathbf{g}'(\mathbf{z}) = \text{diag}([g'(u_i)])$, $G(\mathbf{z}) = \int_C \mathbf{g}(\mathbf{u}) \cdot d\mathbf{u}$ and $F(\mathbf{z}) = \int_C \mathbf{f}(\mathbf{u}) \cdot d\mathbf{u}$.

The potential function is defined in the way that $\min\{U(\mathbf{z}; p_e)\} > 0$ for $p_e < p_e^*$. For the unonstrained coupled system on the other hand, we have $U'(\mathbf{z}; p_e) > 0$ for $p_e \leq p_e^\dagger$. In other words, in order to find p_e^*, it is sufficient to find the maximum p_e such that $\min\{U(\mathbf{z}; p_e)\} > 0$ [14]. We will use this finding and compare the two thresholds later in the results section but intuitively, one expects that $p_e^\dagger \leq p_e^*$, which means the side information which constrains the coupled system results in better error correction performance.

The next lemma shows that once $p_e \leq p_e^\dagger$, the probability of error in the unconstrained coupled system, shown in equation (6), converges to zero.

Lemma 2: For $p_e \leq p_e^\dagger$, the fixed point of equation (6) is 0 and is achieved by iterative updates of Algorithm 2.

Proof: The proof is straightforward and results from the defintion of the potential. Because of the defintion of $p_e^\dagger$, we know that for $p_e \leq p_e^\dagger$ we have $U'(\mathbf{z}; p_e) > 0$. From equation (9), we have $U'(\mathbf{z}; p_e) = \mathbf{g}'(\mathbf{z})(\mathbf{z} - A^\top \mathbf{f}(A\mathbf{g}(\mathbf{z}))$. Given that $\mathbf{g}'(\mathbf{z}) > 0$, from $U'(\mathbf{z}; p_e) > 0$ we conclude that $\mathbf{z} - A^\top \mathbf{f}(A\mathbf{g}(\mathbf{z}) > 0$. This is equivalent to saying that

$$z^{(\ell)}(t+1) = A^\top \mathbf{f}(A\mathbf{g}(\mathbf{z}^{(\ell)}(t)) < z^{(\ell)}(t).$$

Therefore,$z^{(\ell)}(t) \to 0, \forall \ell$. ∎

Now to address the performance of the constrained coupled system, we use the results in [14] and [13] in the next theorem.

Theorem 3: For the constrained coupled neural associative memory, when $p_e < p_e^*$ the potential function decreases in each iteration. Furthermore, if $L > \|U"(\mathbf{z}; p_e)\|_\infty / \Delta E(p_e)$, the only fixed point of the recursion (6) is equal to $\mathbf{0}$.

Proof: The proof of the theorem relies on results from [13] to show that the entries in the vector $\mathbf{z}(t) = [z^{(1)}(t), \ldots, z^{(L)}(t)]$ are non-decreasing, i.e. $z^{(1)}(t) \leq z^{(2)}(t) \leq \cdots \leq z^{(L)}(t)$. This can be shown using induction and the the functions $\mathbf{f}(., p_e)$ and $\mathbf{g}(.)$ are non-decreasing (see the proof of Lemma 22 in [13] for more details).

[4]Note that A corresponds to the unconstrained system. A similar matrix can be defined for the constrained case.

Then, one can apply the result of Lemma 3 in [14] to show that the potential function of the constrained coupled system decreases in each iteration. Finally, when $L > \|U"(\mathbf{z}; p_e)\|_\infty / \Delta E(p_e)$ one could apply Theorem 1 of [14] to show the convergence of the probability of errors to zero. ∎

V. Pattern Retrieval Capacity

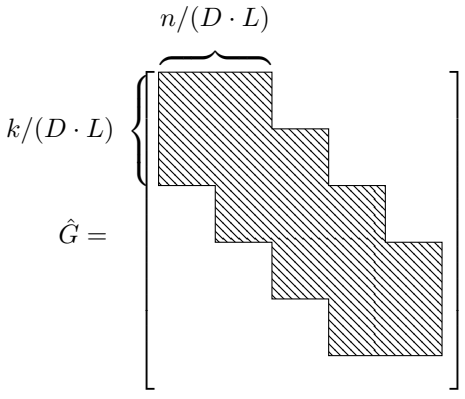
In this section, we prove that the number of patterns that can be memorize by the proposed scheme is exponential in n, the pattern size. To this end, note that the number of patterns that can be memorized is only related to the size of the subspace that sub-patterns come from. In other words, the number of patterns $\mathcal{C}$ which the network memorizes does not depend on the learning or recall algorithms (except its obvious effect on the running time). Therefore, in order to prove that $\mathcal{C}$ could exponentially scale with n, we show there exists such a subspace with exponentially large number of members (in terms of n).

Theorem 4: Let $\mathcal{X}$ be the $\mathcal{C} \times n$ dataset matrix, formed by $\mathcal{C}$ vectors of length n with entries from the set $\mathcal{S}$. Let also $k = rn$ for some $0 < r < 1$. Then, there exists a set of patterns for which $\mathcal{C} = a^{rn}$, with $a > 1$, and rank$(\mathcal{X}) = k < n$.

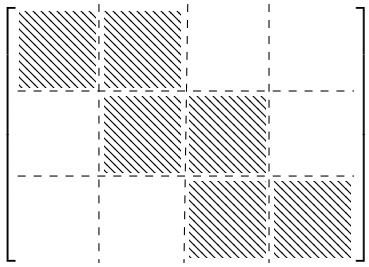
Proof: The proof is based on construction: we construct a data set $\mathcal{X}$ with the required properties such that it can be memorized by the proposed neural network. To simplify the notations, we assume all the clusters have the same number of pattern and constraint neurons, denote by $\tilde{n}_c$ and $\tilde{m}_c$. In other words, $n_{\ell,d} = \tilde{n}_c$ and $m_{\ell,d} = \tilde{m}_c$ for all $\ell = \{1, \ldots, L\}$and $d = \{1, \ldots, D\}$.

We start by considering a matrix $G \in \mathbb{R}^{k \times n}$, with non-negtaive integer-valued entries between 0 and $\gamma - 1$ for some $\gamma \geq 2$. We also assume $k = rn$, with $0 < r < 1$.

To construct the database, we first divide the columns of G into L sets, each corresponding to the neurons in one plain. Furthermore, in order to ensure that all the sub-patterns within each cluster for a supspace with dimension less than $\tilde{n}_c$, we propose the following structure for the generator matrix G. This strucutre ensures that the rank of any sub-matrix of G composed of $\tilde{n}_c$ columns is less than $\tilde{n}_c$. In the matrices below, the hatched blocks represent parts of the matrix with *some* non-zero entries. To simplfiy visualization, let us first define the sub-matrix $\hat{G}$ as the building blocks of G:



Then, G is structured as

$G=$ 

where each hatched block represents a random realization of $\hat{G}$.

Now consider a random vector $u \in \mathbb{R}^k$ with integer-valued-entries between 0 and $\upsilon - 1$, where $\upsilon \geq 2$. We construct the dataset by assigning the pattern $\mathbf{x} \in \mathcal{X}$ to be $\mathbf{x} = \mathbf{u} \cdot G$, *if* all the entries of $\mathbf{x}$ are between 0 and $S-1$. Obviously, since both $\mathbf{u}$ and G have only non-negative entries, all entries in $\mathbf{x}$ are non-negative. Therefore, it is the $S-1$ upper bound that we have to worry about.

Let ϱ_j denote the j^{th} column of G. Then the j^{th} entry in $\mathbf{x}$ is equal to $x_j = \mathbf{u} \cdot \varrho_j$. Suppose ϱ_j has d_j non-zero elements. Then, we have:

$$x_j = u \cdot \varrho_j \leq d_j(\gamma - 1)(\upsilon - 1)$$

Therefore, letting $d^* = \max_j d_j$, we could choose γ, υ and d^* such that

$$S - 1 \geq d^*(\gamma - 1)(\upsilon - 1) \tag{10}$$

to ensure all entries of x are less than S.

As a result, since there are υ^k vectors u with integer entries between 0 and $\upsilon-1$, we will have $\upsilon^k = \upsilon^{rn}$ patterns forming $\mathcal{X}$. Which means $\mathcal{C} = \upsilon^{rn}$, which would be an exponential number in n if $\upsilon \geq 2$. ∎

As a practical example, one could select G and u to have $0/1$ entries (i.e. $\gamma = \upsilon = 2$) and in G, $d^* = 10$. Then it is sufficient to have $S \geq 11$ to have a pattern retrieval capacity of $\mathcal{C} = 2^{rn}$.

Finally, note that inequality (10) corresponds to the worst-case scenario in constructing the patterns in the sense even if not all the patterns $\mathbf{x} = \mathbf{u} \cdot G$ satisfy the bound on maximum value of entries, still a polynomially many of them satisfy this requirement. These vectors correspond to the message vectors $\mathbf{u}$ that are "sparse" as well, i.e. do not have all entries greater than zero. In this case, although one will not achieve the exponential scaling, it is possible to reach polynomial capacities which is still much better than the linear scaling in Hopfield networks [1].

VI. Simulations

A. Simulation Scenario

The first step in designing a neural associative memory is to learn the connectivity matrices of clusters $W^{(\ell,d)}$ for $\ell = 1, \ldots, L$ and $d = 1, \ldots, D$. One can accomplish this objective by applying, for instance, the learning algorithms proposed in [16] to the database in hand.

However, since in this paper we are mainly concerned with the performance of the recall algorithm, we assume the learning phase is done and therefore we have the weighted connectivity graphs available. In the simulations we produce these matrices by generating sparse random bipartite graphs and assign random weights to the connections.

We treat the patterns in the database as $2D$ images of size 64×64. Given the weight matrices and the fact that they are orthogonal to the sub-pattern, we could assume w.l.o.g that in the recall phase we are interested in recalling the all-zero pattern from a noisy version. Therefore, during the recall phase we only generate some random noise and attempt to eliminate the noise using the Algorithm 2.

In the scenario considered in this section, we have generated a random network with 29 planes and 29 clusters within each plane (i.e. $L = D = 29$). Each local cluster was composed of 8×8 neurons and each pattern neuron (pixel) was connected to 2 consecutive planes and 2 clusters within each plane (except at the boundaries), obtained by moving the 8×8 rectangular window over the $2D$ pattern horizontally and vertically (see Figure 1).

We investigated the performance of the recall phase by randomly generating a $2D$ noise pattern in which each entry is set to ± 1 with probability $p_e/2$ and 0 with probability $1 - p_e$. We then apply Algorithm 2 to eliminate the noise. Once finished, we declare an error if the output of the algorithm, $\hat{\mathbf{x}}$, is not equal to the the pattern $\mathbf{x}$, which is assumed to be the all-zero vector.

B. Results

Figure 4 illustrates the final error rate of the proposed algorithm, for the constrained and uncosntrained system. To obtain the final recall error probability, we have repeated Algorithm 2 for $t_{\max} = 10$ times. For the constrained system, we fixed the state of a patch of neuron on size 3×3 at the four corners of the $2D$ pattern. In the same figure, the results are also compared to the similar algorithms in [5] and [11]. In [5], there are no clustering while in [11] the network is divided into some *non-overlapping* clusters with a second nerual level to compensate for the lack of collaboration among non-overlapping clusters. As obvious from the figure, the performance of the proposed algorithms in this paper is significantly better than those in [5] and [11].[5]

Table I shows the thresholds p_e^* and $p_e^\dagger$ for the constrained and unconstrained systems, respectively. Note that the thresholds are given for two different setups. Letting e be the minimum number of error that Algorithm 1 can correct with high probability, the two setups in Table I corresponds to the cases where $e = 1$ and $e = 0$.

Given that from Theorem 1, each cluster can correct a single error with probability at least $1 - \left(\frac{\bar{d}}{m}\right)^{d_{min}}$, we expect that the threshold obtained in practice lie somewhere between the corresponding thresholds for the two mentioned setups.

[5]In Figure 4, the performance of the algorithm in [11] is a bit worse than that of [5]. That is because in this paper, we generated the graph of [5] at random while the neural graph of [11] was obtained by a learning algorithm resulting in less efficient topologies.

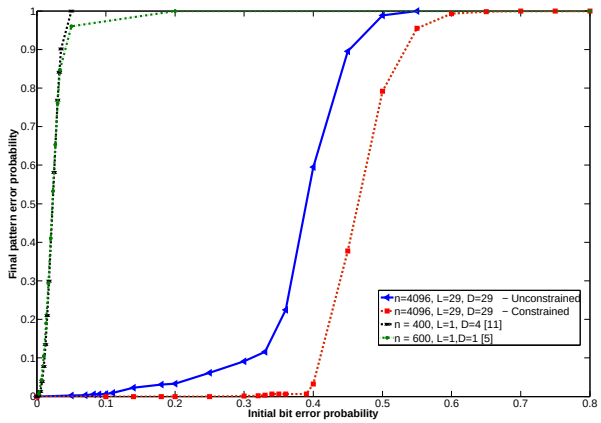


Fig. 4: The final pattern error probability for the constrained and unconstrained coupled neural systems.

	$p_e^\dagger$	p_e^*
$e=1$	0.197	0.369
$e=0$	0.078	0.114

TABLE I: The thresholds on initial probability of error for two performance extremes of Algorithm 1

Comparing the result of Figure 4 and that of Table I, confirms our expectation. From Figure 4 we notice that $0.199 \leq p_e^* \simeq 0.32 \leq 0.394$. As for $p_e^\dagger \simeq .07$ which is close to the lower bound on the threshold. The mismatch might be attributed to the fact that the derived bounds are usually valid in the limit of large n and L.

VII. Conclusions and Future Works

In this paper, we proposed a novel architecture for neural associative memories. The proposed architecture comprises a set of neural planes with sparsely connected overlapping clusters. Furthermore, planes are sparsely connected together as well. Given the similarity of the suggested framework to spatially-coupled codes, we employed recent developments in analyzing these codes to theoretically investigate the performance of the proposed neural algorithm. We derived two thresholds on the maximum initial bit error probability that can be corrected by the proposed algorithm with probability close to 1. Using simulations, we confirmed that there is a good match between the thresholds derived theoretically and those obtained in practice.

Given that our main interest in this paper was the performance of the error correcting algorithm in the recall phase, we did not address the learning phase here. However, we are currently in the middle of applying the learning method in [16] to a database of natural images to assess the performance of the recall algorithm in this real-world setup as well. Preliminary results are encouraging in the sense that they indicate when a suitable feature map is provided, one can apply the learning and recall algorithms to these features in order to implement a neural associative memory over the set of features with good error correction capabilities. Further investigation in finding a good feature map is required though.

Acknowledgments

The authors would like to thank Prof. Henry D. Pfister, Mr. Vahid Aref and Mr. Seyed Hamed Hassani for their helpful comments and discussions.

References

[1] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. Natl. Acad. Sci., Vol. 79, 1982, pp. 2554-2558.

[2] S. S. Venkatesh, D. Psaltis, *Linear and logarithmic capacities in associative neural networks*, IEEE Trans. Inf. Theory, Vol. 35, No. 3, 1989, pp. 558-568.

[3] S. Jankowski, A. Lozowski, J.M., Zurada, *Complex-valued multistate neural associative memory*, IEEE Tran. Neur. Net., Vol. 1 , No. 6, 1996, pp. 1491-1496.

[4] M. K. Muezzinoglu, C. Guzelis, J. M. Zurada, *A new design method for the complex-valued multistate Hopfield associative memory*, IEEE Trans. Neur. Net., Vol. 14, No. 4, 2003, pp. 891-899.

[5] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Proc. IEEE Information Theory Workshop, 2011.

[6] D. O. Hebb, *The organization of behavior*, New York: Wiley & Sons, 1949.

[7] R. McEliece, E. Posner, E. Rodemich, S. Venkatesh, *The capacity of the Hopfield associative memory*, IEEE Trans. Inf. Theory, Jul. 1987.

[8] V. Gripon, C. Berrou, *Sparse neural networks with large learning diversity*, IEEE Trans. on Neural Networks, Vol. 22, No. 7, 2011, pp. 10871096.

[9] C. Berrou, V. Gripon, *Coded Hopfield Networks*, Proc. Symp. on Turbo Codes and Iterative Information Processing, pp. 15, 2010.

[10] P. Peretto, J. J. Niez, *Long term memory storage capacity of multiconnected neural networks*, Biological Cybernetics, Vol. 54, No. 1, 1986, pp. 53-63.

[11] A. H. Salavati, A. Karbasi, *Multi-Level Error-Resilient Neural Networks*, IEEE Int. Symp. Inf. Theory (ISIT 2012), 2012.

[12] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the theory of neural computation*, USA: Addison-Wesley, 1991.

[13] S. Kudekar, T. Richardson, R. Urbanke, *Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC*, IEEE Trans. Inf. Theory, Vol. 57, No. 2, 2012, pp. 803-834.

[14] A. Yedla, Y. Jian, P. S. Nguyen, H. D. Pfister, *A simple proof of threshold saturation for coupled scalar recursions*, To appear in Int. Symp. Turbo codes and Itr. Info. Processing (ISTC), 2012

[15] Y. Jian, H. D. Pfister, K. R. Narayanan, *Approaching capacity at high rates with iterative hard-decision decoding*, Int. Symp. Inf. Theory (ISIT), 2012.

[16] A. Karbasi, A. H. Salavati, A. Shokrollahi, *A simple algorithm to learn sparse dual vectors using a neural network*, Technical report, 2012, EPFL (available at $http://algo.epfl.ch/\ amir/Learning.pdf$

[17] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, R. Singh "Cognitive computing," Communications of the ACM, Vol. 54, No. 8, 2011, pp. 62-71.

[18] P. Dayan, L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*, MIT Press, 2004.