

Coupled Neural Associative Memories

Amin Karbasi, Amir Hesam Salavati, and Amin Shokrollahi
School of Computer and Communication Sciences
Ecole Polytechnique Federale de Lausanne (EPFL)
Switzerland

Abstract

We propose a novel architecture to design a neural associative memory that is capable of learning a large number of patterns and recalling them later in presence of noise. It is based on dividing the neurons into local clusters and parallel plains, very similar to the architecture of the visual cortex of macaque brain. The common features of our proposed architecture with those of spatially-coupled codes enable us to show that the performance of such networks in eliminating noise is drastically better than the previous approaches while maintaining 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. We also present computational experiments that lend additional support to the theoretical analysis.

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.

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. This problem, in its core, is very similar to the 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. 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 central roles.

Despite these similarities in the objectives and 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 ($0 < r < 1$). This is achieved by intelligently introducing redundancy among the transmitted messages, which is later used to recover the correct pattern from the received noisy version. In contrast, until recently, artificial neural associative memories were only capable of memorizing $O(n)$ binary patterns of length n (see, [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 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 common *structure*, namely, patterns all belong to a subspace with dimension $k < n$. Karbasi et al. [6] extended this model to "modular" neural architectures and introduced a suitable online learning algorithm. They showed that the modular structure improves the noise tolerance properties significantly.

In this work, we extend the model of [6] further by linking the modular structures to obtain a "coupled" neural architecture. Interestingly, this model looks very similar to some models for processing visual signals in the macaque brain [7]. We then make use of the recent developments in the analysis of spatially-coupled codes by [8] and [9] to derive analytical bounds on the performance of the proposed method. Finally, using simulations we show that the proposed method achieves much better performance measures compared to previous work in eliminating noise during the recall phase.

II. RELATED WORK

Arguably, one of the most influential models for neural associative memories was introduced by Hopfield [1]. A "Hopfield network" is a complete graph of n neurons that memorizes a subset of *randomly* chosen binary patterns of length n . It is known that the pattern retrieval capacity (i.e., maximum number of memorized patterns) of Hopfield networks is $\mathcal{C} = (n/2 \log(n))$ [10].

There have been many attempts to increase the pattern retrieval capacity of such networks by introducing offline learning schemes (in contrast to online schemes) [2] or multi-state neurons (instead of binary) [4], all of which resulted in memorizing at most $O(n)$ patterns.

By dividing the neural architecture into smaller *disjoint* blocks, Venkatesh [11] increased the capacity to $\Theta(b^{n/b})$ (for *random* patterns), where $b = \omega(\ln n)$ is the size of blocks. This is a huge improvement but comes at the price of limited *worst-case* noise tolerance capabilities. Specifically, due to the non-overlapping nature of the clusters (blocks), the error correction is limited by the performance of individual clusters as there is no inter-cluster communication. With *overlapping* clusters, one could hope for achieving better error correction, which is the reason we consider such structures in this paper.

More recently, a new perspective has been proposed with the aim of memorizing only those patterns that possess some degree of 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, Gripon and Berrou [12] proposed a 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. Along the same lines, by considering patterns that lie in a subspace of dimension $k < n$, Kumar et al. were able to show an exponential scaling in the pattern retrieval capacity, i.e., $C = O(a^n)$, with some $a > 1$. This model was later extended to modular patterns, i.e., those in which patterns are divided into sub-patterns where each sub-pattern comes from a subspace [6]. The authors provided a simple iterative learning algorithm that demonstrates a better performance in the recall phase as compared to [5].

In this paper, we follow the same line of thought by linking several instances of the model proposed in [6] in order to have a "coupled" structure. More specifically, the proposed model is based on *overlapping* local clusters, arranged in parallel planes, with neighboring neurons. At the same time, we enforce sparse connections between various clusters in different planes. The aim is to memorize only those patterns for which local sub-patterns in the domain of each cluster show a certain degree of redundancy. On the one hand, the obtained model looks similar to neural modules in the visual cortex of the macaque brain [7]. And on the other hand, it is similar to spatially-coupled codes on graphs [9]. Specifically, our suggested model is closely related to the spatially-coupled Generalized LDPC code (GLDPC) with Hard Decision Decoding (HDD) proposed in [13]. This similarity helps us borrow analytical tools developed for analyzing such codes [8] and investigate the performance of our proposed neural error correcting algorithm.

The proposed approach enjoys the simplicity of message passing operations performed by neurons as compared to the more complex iterative belief propagation decoding procedure of spatially coupled codes [9]. This simplicity may lead to an inferior performance but already allows us to outperform the prior error resilient methods suggested for neural associative memories in the literature.

III. PROBLEM SETTING AND NOTATIONS

In this paper, we work with non-binary neural networks, where the state of each neuron is a bounded non-negative integer (which can be thought of as the short-term firing rate of neurons in a real neural network). Like other neural networks, neurons could only perform simple operations, i.e. *linear summation* and *non-linear thresholding*. More specifically, a neuron x updates its state based on the states of its neighbors $\{s_i\}_{i=1}^n$ as follows:

- 1) It computes the weighted sum $h = \sum_{i=1}^n w_i s_i$, where w_i denotes the weight of the input link from s_i .
- 2) It updates its state as $x = f(h)$, where $f: \mathbb{R} \rightarrow \mathcal{S}$ is a possibly non-linear function.

Let \mathcal{X} denote a dataset of \mathcal{C} patterns of length n where the patterns are integer-valued with entries in $\mathcal{S} = \{0, \dots, S-1\}$. A natural way of interpreting this model is to consider the entries as the short-term firing rate of n neuron. In this paper we are interested in designing a neural network that is capable of memorizing these patterns in such a way that later, and in response to noisy queries, the correct pattern will be returned. To this end, we break the patterns into smaller pieces/sub-patterns and learn the resulting sub-patterns separately (and in parallel). Furthermore, as our objective is to memorize those patterns that are highly correlated, we only consider a dataset in which the sub-patterns belong to a subspace (or have negligible minor components).

More specifically, and to formalize the problem in a way which is similar to the literature on spatially coupled codes, we divide each pattern into L sub-patterns of the same size and refer to them as *planes*. Within each plane, we further divide the patterns into D *overlapping* clusters, i.e., an entry in a pattern can lie in the domain of multiple clusters. We also assume that each element in plane ℓ is connected to at least one cluster in planes $\ell - \Omega, \dots, \ell + \Omega$ (except at the boundaries). Therefore, each entry in a pattern is connected to $2\Omega + 1$ planes, on average.

An alternative way of understanding the model is to consider 2D datasets, i.e., images. In this regard, each row of the image corresponds to a plane and clusters are the overlapping "receptive fields" which cover an area composed of neighboring pixels in different rows (planes). This is in fact very similar to the configuration of the receptive fields in human visual cortex [14]. Our assumptions on strong correlations then translates into assuming strong linear dependencies within each receptive field for all patterns in the dataset.

Noise model: Throughout this paper, we consider an additive noise model. More specifically, the noise is an integer-valued vector of size n and for simplicity we assume that its entries are $\{-1, 0, +1\}$, where a -1 (resp. $+1$) corresponds to a neuron

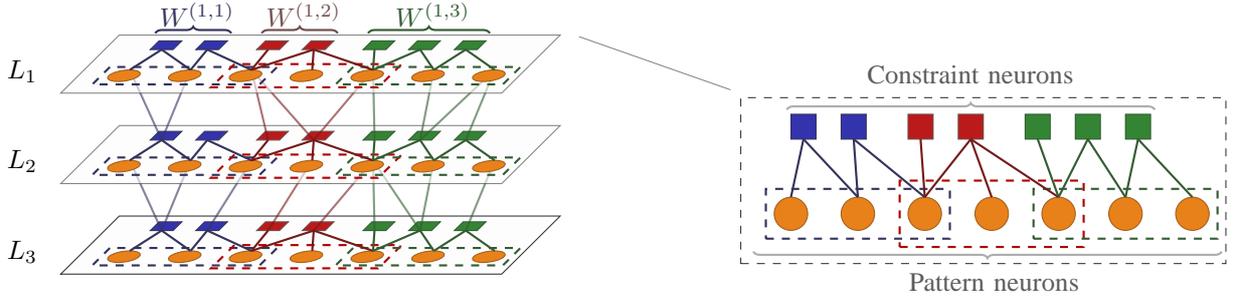


Fig. 1: A coupled neural associative memory.

skipping a spike (resp. fire one more spike than expected).¹ The noise probability is denoted by p_e and each entry of the noise vector is $+1$ or -1 with probability $p_e/2$.²

Pattern Retrieval Capacity: This is the maximum number of patterns that can be memorized by a network while still being able to return reliable responses in the recall phase.

IV. LEARNING PHASE

To "memorize" the patterns, we learn a set of vectors that are orthogonal to the sub-patterns in each cluster, using the algorithm proposed in [6]. The output of the learning algorithm is an $m_{\ell,d} \times n_{\ell,d}$ matrix $W^{(\ell,d)}$ for cluster d in plane ℓ . The rows of this matrix correspond to the dual vectors and the columns correspond to the corresponding entries in the patterns. Therefore, by letting $\mathbf{x}^{(\ell,d)}$ denote the sub-pattern corresponding to the domain of cluster d of plane ℓ , we have

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

These matrices (i.e., $W^{(\ell,d)}$) form the connectivity matrices of our neural graph, in which we can consider each cluster as a bipartite graph composed of *pattern* and *constraint* neurons. The left panel of Figure 1 illustrates the model, in which the circles and rectangles correspond to pattern and constraint neurons, respectively. The details of the first plane are magnified in the right panel of Figure 1.

Cluster d in plane ℓ contains $m_{\ell,d}$ constraint neurons and is connected to $n_{\ell,d}$ pattern neurons. The constraint neurons do not have any overlaps (i.e. each one belongs only to one cluster) whereas the pattern neurons can have connections to multiple clusters and planes. To ensure good error correction capabilities we aim to keep the neural graph sparse (this model shows significant similarity to some neural architectures in the macaque brain [7]).

We also consider the overall connectivity graph of plane ℓ , denoted by $\widetilde{W}^{(\ell)}$, 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 (unweighted) edge to the corresponding super node. Figure 2 illustrates this graph for plane 1 in Figure 1.

V. RECALL PHASE

The main goal of our architecture is to retrieve correct memorized patterns in response to noisy queries. At this point, the neural graph is learned (fixed) and we are looking for a simple iterative algorithm to eliminate noise from queries. The

¹Other noise models, such as real-valued noise, can be considered as well. However, the thresholding function $f : \mathbb{R} \rightarrow \mathcal{S}$ will eventually lead to integer-valued "equivalent" noise in our system.

²Our algorithm can also deal with erasures. Note that an erasure at node x_i corresponds to an integer noise with the negative value of x_i . So once we have established the performance of our algorithm for integer-valued noise, it would be straightforward to extend the algorithm to erasure noise models.

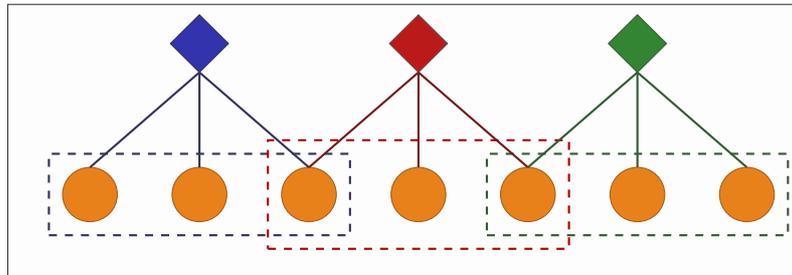


Fig. 2: A connectivity graph with neural planes and super nodes. It corresponds to plane 1 of Fig. 1.

Algorithm 1 Error Correction Within Cluster [16]

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

Output: Correct memorized sub-pattern $\mathbf{x}^{(\ell,d)}$.

1: **for** $t = 1 \rightarrow t_{\max}$ **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: **end for**

Algorithm 2 Error Correction of the Coupled Network

Input: Connectivity matrix $(W^{(\ell,d)}, \forall \ell, \forall d)$, iteration t_{\max}

Output: Correct memorized pattern $\mathbf{x} = [x_1, x_2, \dots, x_n]$

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

2: **for** $\ell = 1 \rightarrow L$ **do**

3: **for** $d = 1 \rightarrow D$ **do**

4: Apply Algorithm 1 to cluster d of neural plane ℓ .

5: Update the value of pattern nodes $\mathbf{x}^{(\ell,d)}$ only if all the constraints in the clustered are satisfied.

6: **end for**

7: **end for**

8: **end for**

proposed recall algorithm in this paper is the extension of the one in [6] to the coupled neural networks. For the sake of completeness, we briefly discuss the details of the approach suggested in [6] and explain the extension subsequently.

The recall method in [6] is composed of two types of separate algorithms: *local* (or intra-cluster) and *global* (or inter-cluster). The local algorithm tries to correct errors within each cluster by the means of simple message-passings. It relies on 1) pattern neurons transmitting their state to the constraint neurons and then on 2) constraint neurons checking if the constraints are met (i.e. the values transmitted by the pattern nodes to the constraint nodes should sum to zero). If not, the constraint neurons send a message telling the direction of the violation (i.e. if the input sum is less or greater than zero). The pattern neurons then updates their state according to the received feedback from their neighboring constraint neurons on a majority voting basis. The process is summarized in Algorithm 1.

The overall error correction properties of Algorithm 1 is fairly limited. In fact, it can be shown that in a given cluster, the algorithm could correct a single input error (i.e only one pattern neurons deviating from its correct state) with probability $1 - (\bar{d}/m)^{d_{\min}}$, where \bar{d} and d_{\min} are the average and minimum degree of the pattern nodes. For more than one input error, the algorithm can easily get stuck. To overcome this drawback, Karbasi et al. [6] proposed a sequential procedure by applying Algorithm 1 in a Round Robin fashion to each cluster. If the errors were eliminated, the pattern nodes in the cluster keep their new values, and revert back to their original states, otherwise. This scheduling technique is in esprit similar to the Peeling Algorithm widely used in LDPC codes [15]. Correcting the error in the clusters with a single error can potentially help the neighboring clusters.

Inspired by this boost in the performance, we can stretch the error correction capabilities even further by coupling several neural "planes" with many clusters together, as mentioned earlier. We need to modify the global error correcting algorithm in such a way that it first acts upon the clusters of a given plane in each round before moving to the next plane. The whole process is repeated few times until all errors are corrected or a threshold on the number of iterations is reached (t_{\max}). Algorithm 2 summarizes our approach.

VI. PERFORMANCE ANALYSIS

We consider two variants of the above error correction algorithm. In the first one, called *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 to known and correct states, similar to spatially-coupled codes [8], [9]. In the case of neural associative memory, the side information can come from the context. For instance, when trying to correct the error in the sentence "The cat flies", we can use the side information (flying) to guess the correct answer among multiple choices. Without this side information, we cannot tell if the correct answer corresponds to *bat* or *rat*.³

In the other variant, called *unconstrained* coupled neural error correction, we perform the error correction without providing any side information. This is similar to many standard recall algorithms in neural networks. In fact, the unconstrained model can be thought of as a very large convolutional network similar to the model proposed in [6]. Thus, the unconstrained model serves as a benchmark to evaluate the performance of the proposed coupled model in this paper.

Let $z^{(\ell)}(t)$ denote the *average* probability of error for pattern nodes across neural plane ℓ in iteration t . Thus, a super constraint node in plane ℓ receives noisy messages from its neighbors with an average probability $\bar{z}^{(\ell)}$:

$$\bar{z}^{(\ell)} = \frac{1}{2\Omega + 1} \sum_{j=-\Omega}^{\Omega} z^{(\ell-j)} \text{ s.t. } z^{(\ell)} = 0, \forall \ell \notin \{1, \dots, L\}.$$

Our goal is to derive a recursion for $z^{(\ell)}(t+1)$ in terms of $z^{(\ell)}(t)$ and $\bar{z}^{(\ell)}(t)$. To this end, in the graph $\widetilde{W}^{(\ell)}$ let $\lambda_i^{(\ell)}$ and $\rho_j^{(\ell)}$ be the fraction of edges (normalized by the total number of edges in graph $\widetilde{W}^{(\ell)}$) connected to pattern and super nodes with degree i and j , respectively. We define the degree distribution polynomials in plane ℓ from an *edge perspective* as $\lambda^{(\ell)}(x) = \sum_i \lambda_i^{(\ell)} x^i$ and $\rho^{(\ell)}(x) = \sum_j \rho_j^{(\ell)} x^{j-1}$.

Lemma 1. *Let us define $g(z) = 1 - \rho(1-z) - \sum_{i=1}^{e-1} \frac{z^i}{i!} \frac{d^i \rho(1-z)}{dz^i}$ and $f(z; p_e) = p_e \lambda(z)$, where e is the number of errors each cluster can correct. Then,*

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

Proof: Without loss of generality, we prove the lemma for the case that each cluster could correct at least two errors with high probability, i.e. $e = 2$. Extending the proof to $e > 2$ would be straightforward.

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 *super constraint node* in plane ℓ sending an erroneous message to its neighbors. We will derive recursive expressions for $z^{(\ell)}(t)$ and $\pi^{(\ell)}(t)$.

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

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

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

Let $\pi_i^{(\ell)}$ denote the probability that a super constraint node with degree i in plane ℓ sends an erroneous message to one of its neighboring noisy pattern nodes. Then, knowing that each super constraint node (cluster) is capable of correcting at least $e = 2$ errors, $\pi_i^{(\ell)}$ is equal to the probability of receiving two or more noisy messages from *other* pattern neurons,

$$\pi_i^{(\ell)} = 1 - \left(1 - \bar{z}^{(\ell)}\right)^{i-1} - (i-1)\bar{z}^{(\ell)} \left(1 - \bar{z}^{(\ell)}\right)^{i-2}.$$

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

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

where $\rho(z) = \sum_i \rho_i z^{i-1}$ is the super constraint 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

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

³The same situation also happens in dealing with erasures, i.e. when trying to fill in the blank in the sentence "The at flies".

Now consider a given pattern neuron with degree j in plane ℓ . Let $z_j^{(\ell)}(t+1)$ denote the probability of sending an erroneous message by this node in iteration $t+1$. Then, $z_j^{(\ell)}(t+1)$ is equal to the probability of this node being noisy in the first place (p_e) and having all its super constraint 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

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

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:

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

The decoding 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

$$f\left(\frac{1}{2\Omega+1} \sum_{i=-\Omega}^{\Omega} g(\bar{z}^{(\ell-i)}(t)); p_e\right) < z^{(\ell)} \text{ for } z^{(\ell)} \in [0, p_e].$$

Let p_e^\dagger and p_e^* be the maximum p_e 's that admit successful decoding for the uncoupled and coupled systems, respectively. To estimate these thresholds, we follow the approach recently proposed in [8] and define a potential function to track the evolution of Eq. (2). Let $\mathbf{z} = \{z^{(1)}, \dots, z^{(L)}\}$ denote the vector of average probabilities of error for pattern neurons in each plane. Furthermore, let $\mathbf{f}(\mathbf{z}; p_e) : \mathbb{R}^L \rightarrow \mathbb{R}^L$ and $\mathbf{g}(\mathbf{z}) : \mathbb{R}^L \rightarrow \mathbb{R}^L$ be two component-wise vector functions 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 in Lemma 1. Using these definitions, we can rewrite Eq. (2) in the vector form as [8]:

$$\mathbf{z}(t+1) = A^\top \mathbf{f}(\mathbf{A}\mathbf{g}(\mathbf{z}(t)); p_e) \quad (3)$$

where A is the *coupling matrix* defined as⁴:

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

At this point, the potential function of the unconstrained coupled system could be defined as [8]:

$$\begin{aligned} U(\mathbf{z}; p_e) &= \int_C \mathbf{g}'(\mathbf{u})(\mathbf{u} - A^\top \mathbf{f}(\mathbf{A}\mathbf{g}(\mathbf{u}))). d\mathbf{u} \\ &= \mathbf{g}(\mathbf{z})^\top \mathbf{z} - G(\mathbf{z}) - F(\mathbf{A}\mathbf{g}(\mathbf{z}); p_e) \end{aligned} \quad (4)$$

⁴Matrix A corresponds to the unconstrained system. A similar matrix can be defined for the constrained case.

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

A similar quantity can be defined for the uncoupled (scalar) system as $U_s(z; p_e) = zg(z) - G(z) - F(g(z); p_e)$ [8], where z is the average probability of error in pattern neurons. The scalar potential function is defined in the way that $U'_s(z; p_e) > 0$ for $p_e \leq p_e^\dagger$. In other words, it ensures that $z(t+1) = f(g(z(t); p_e)) < z(t)$ (successful decoding) for $p_e \leq p_e^\dagger$.

Furthermore, let us define $p_e^* = \sup\{p_e \mid \min(U_s(z; p_e)) \geq 0\}$. Thus, in order to find p_e^* , it is sufficient to find the maximum p_e such that $\min\{U_s(z; p_e)\} > 0$ [8]. We will show that the constrained coupled system achieves successful error correction for $p_e < p_e^*$. Intuitively, we expect to have $p_e^\dagger \leq p_e^*$ (side information only helps), and as a result a better error correction performance for the constrained system. Theorem 2 and our experimental result will confirm this intuition later in the paper.

Let $\Delta E(p_e) = \min_z U_s(z; p_e)$ be the *energy gap* of the uncoupled system for $p_e \in (p_e^\dagger, 1]$ [8]. The next theorem borrows the results of [8] and [9] to show that the constrained coupled system achieves successful error correction for $p_e < p_e^*$.

Theorem 2. *In the constrained system, when $p_e < p_e^*$ the potential function decreases in each iteration. Furthermore, if $\Omega > \frac{\|U''(\mathbf{z}; p_e)\|_\infty}{\Delta E(p_e)}$, the only fixed point of Eq. (3) is 0.*

Proof: The proof of the theorem relies on results from [9] to show that the entries in the vector $\mathbf{z}(t) = [z^{(1)}(t), \dots, z^{(L)}(t)]$ are non-decreasing, i.e.,

$$z^{(1)}(t) \leq z^{(2)}(t) \leq \dots \leq z^{(L)}(t).$$

This can be shown using induction and the fact that the functions $\mathbf{f}(\cdot, p_e)$ and $\mathbf{g}(\cdot)$ are non-decreasing (see the proof of Lemma 22 in [9] for more details).

Then, one can apply the result of Lemma 3 in [8] to show that the potential function of the constrained coupled system decreases in each iteration. Finally, when

$$\Omega > \|U''(\mathbf{z}; p_e)\|_\infty / \Delta E(p_e)$$

one could apply Theorem 1 of [8] to show the convergence of the probability of errors to zero. \blacksquare

Note that Theorem 2 provides a sufficient condition (on Ω) for the coupled system to ensure it achieves successful error correction for every p_e upto $p_e = p_e^*$. However, the condition provided by Theorem 2 usually requires Ω to be *too large*, i.e. Ω is required to be as large as 1000 to 10000, depending on the degree distributions. Nevertheless, in the next section we show that the analysis is still quite accurate for moderate values of Ω , i.e. $\Omega \simeq 2, 3$, meaning that a system with a small coupling parameters could still achieve very good error correction in practice.

VII. PATTERN RETRIEVAL CAPACITY

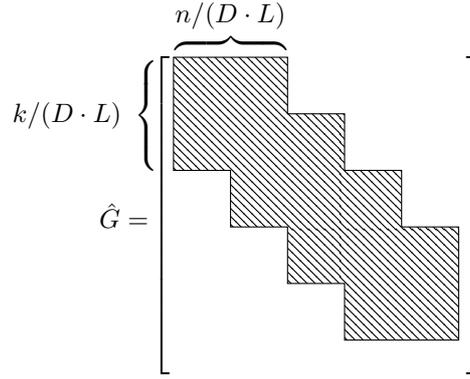
The following theorem shows that the number of patterns that can be memorized by the proposed scheme is exponential in n , the pattern size.

Theorem 3. *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 $\text{rank}(\mathcal{X}) = k < n$.*

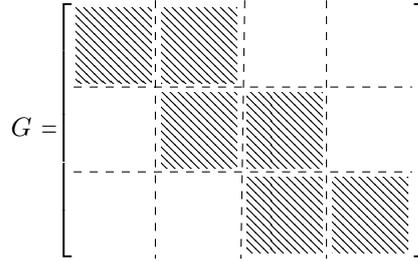
Proof: The proof is based on construction: we construct a data set \mathcal{X} with the required properties such that it can be memorized by the proposed neural network. To simplify the notations, we assume all the clusters have the same number of pattern and constraint neurons, denote by \tilde{n}_c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

We start by considering a matrix $G \in \mathbb{R}^{k \times n}$, with non-negative 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 form a subspace with dimension less than \tilde{n}_c , we propose the following structure for the generator matrix G . This structure 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 simplify visualization, let us first define the sub-matrix \hat{G} as the building blocks of G :



Then, G is structured as



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 $v - 1$, where $v \geq 2$. 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:

$$x_j = \mathbf{u} \cdot \varrho_j \leq d_j(\gamma - 1)(v - 1)$$

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

$$S - 1 \geq d^*(\gamma - 1)(v - 1) \quad (5)$$

to ensure all entries of x are less than S .

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

VIII. SIMULATIONS

In this paper, we are mainly interested in the performance of the recall phase and demonstrate a way, by the means of spatial coupling, to improve upon the previous art. To this end, we assume that the learning phase is done (by using our proposed algorithm in [6]) and we have the weighted connectivity graphs available. For the ease of presentation, we can simply produce these matrices by generating sparse random bipartite graphs and assign random weights to the connections. Given the weight matrices and the fact that they are orthogonal to the sub-patterns, we can assume w.l.o.g that in the recall phase we are interested in recalling the all-zero pattern from its noisy version.

We treat the patterns in the database as $2D$ images of size 64×64 . More precisely, we have generated a random network with 29 planes and 29 clusters within each plane (i.e., $L = D = 29$). Each local cluster is composed of 8×8 neurons and each pattern neuron (pixel) is connected to 2 consecutive planes and 2 clusters within each plane (except at the boundaries). This is achieved by moving the 8×8 rectangular window over the $2D$ pattern horizontally and vertically. The degree distribution of this setting is $\lambda = \{0.0011, 0.0032, 0.0043, 0.0722, 0, 0.0054, 0, 0.0841, 0.0032, 0, 0, 0.098, 0, 0, 0, 0.7284\}$, $\rho_{64} = 1$ and $\rho_j = 0$ for $1 \leq j \leq 63$.

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 with $t_{\max} = 10$ to eliminate the noise.

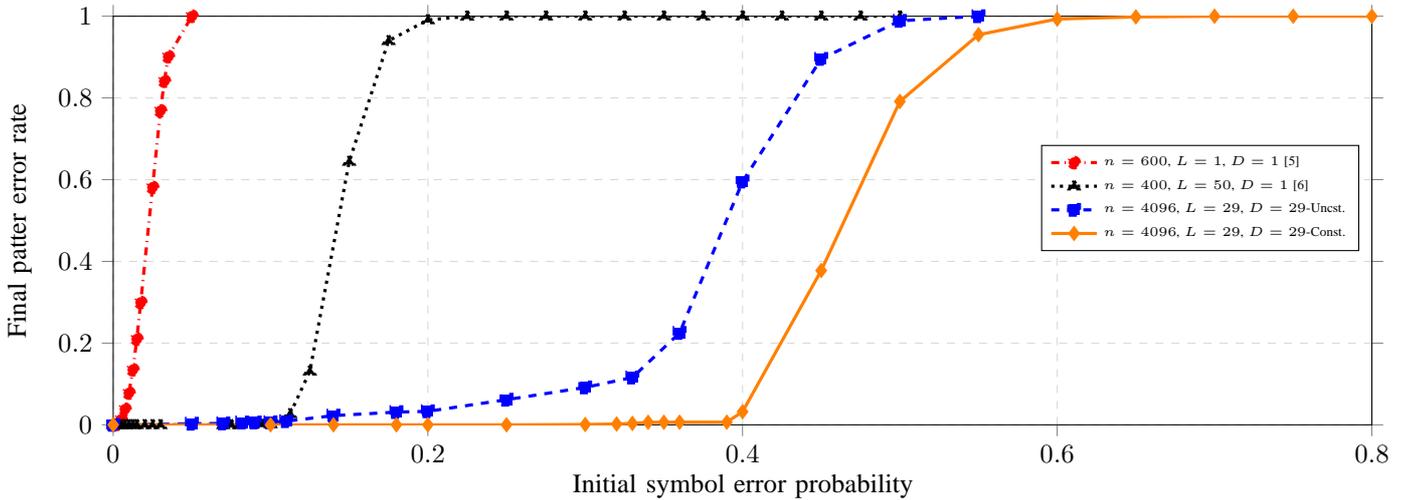


Fig. 3: The final pattern error probability for the constrained and unconstrained coupled neural systems.

Once finished, we declare failure if the output of the algorithm, $\hat{\mathbf{x}}$, is not equal to the pattern \mathbf{x} (assumed to be the all-zero vector).

Figure 3 illustrates the final error rate of the proposed algorithm, for the constrained and unconstrained system. For the constrained system, we fixed the state of a patch of neurons of size 3×3 at the four corners of the $2D$ pattern. The results are also compared to the similar algorithms in [5] and [6] (uncoupled systems). In [5] (the dashed-dotted curve), there are no clustering while in [6] the network is divided into 50 overlapping clusters all lying on a single plane (the dotted curve). Although clustering improves the performance, it is still inferior than the coupled system with some side information (the solid curve). Even though the same recipe (i.e., Alg. 1) is used in all approaches, the differences in the architectures has a profound effect on the performance. One also notes the sheer positive effect of network size on the performance (the dotted vs. dashed curves).

Table I shows the thresholds p_e^\dagger and p_e^* for different values of e . From Figure 3 we notice that $p_e^* \simeq 0.39$ and $p_e^\dagger \simeq .1$ which is close to the thresholds for $e = 2$ in Table I. Note that according to Theorem 2, a sufficient condition for these thresholds to be exact is for Ω to be very large. However, the comparison between Table I and Figure 3 suggest, one could obtain rather exact results even with Ω being rather small.

	p_e^\dagger	p_e^*
$e = 1$	0.078	0.114
$e = 2$	0.197	0.394

TABLE I: The thresholds for the uncoupled (p_e^\dagger) and coupled (p_e^*) systems.

Figure 4 illustrates how the potential function for uncoupled systems behaves as a function of z and for various values of p_e . Note that for $p_e \simeq p_e^*$, the minimum value of potential reaches zero, i.e. $\Delta_E(p_e^*) = 0$, and for $p_e > p_e^*$ the potential becomes negative for large values of z .

IX. CONCLUSIONS

In this paper, we proposed a novel architecture for neural associative memories. The proposed model 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 investigate the performance of our proposed neural algorithm. We also presented numerical simulations that lend additional support to the theoretical analysis. 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 [6] to a database of natural images to assess the performance of the recall algorithm in this real-world setup as well.

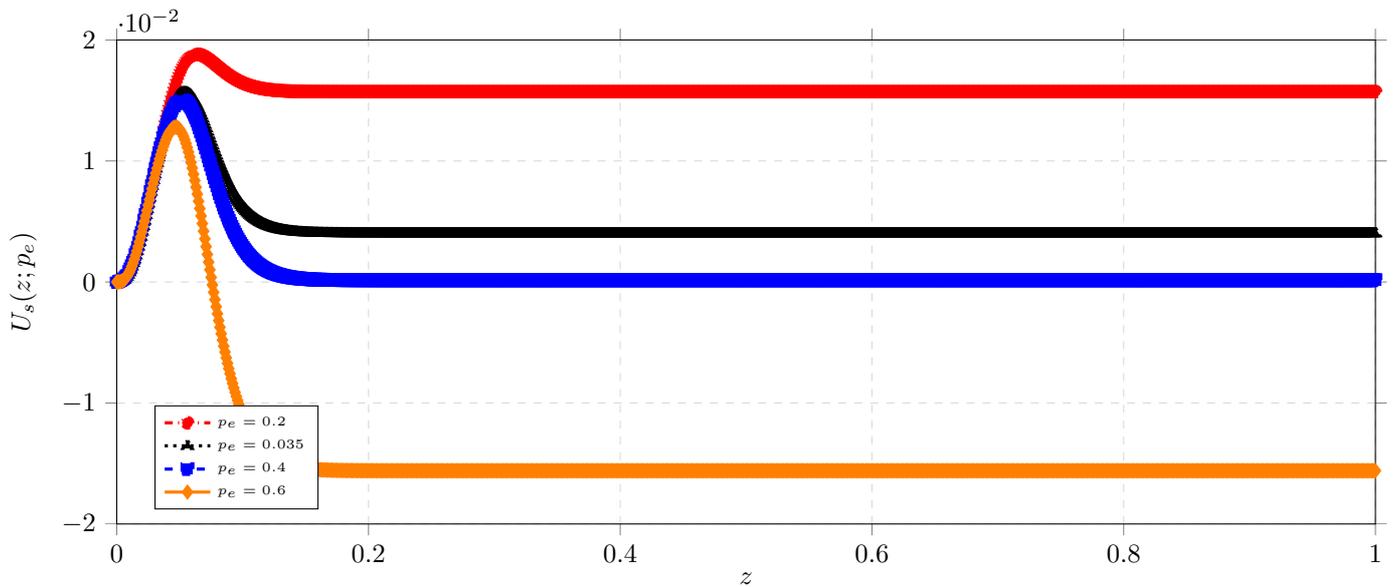


Fig. 4: The scalar potential function U_s as function of average pattern neurons error probability, z , and different initial symbol error probabilities, p_e .

Acknowledgments

The authors would like to thank Prof. Henry D. Pfister, Mr. Vahid Aref, and Dr. 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] A. Karbasi, A. H. Salavati, A. Shokrollahi, *Iterative Learning and Denoising in Neural Associative Memories*, To appear in ICML 2013.
- [7] 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.
- [8] 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
- [9] 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.
- [10] R. McEliece, E. Posner, E. Rodemich, S. Venkatesh, *The capacity of the Hopfield associative memory*, IEEE Trans. Inf. Theory, Jul. 1987.
- [11] S. S. Venkatesh, *Connectivity and Capacity in the Hebb Rule*, in Advances in Neural Networks, (eds. A. Orlicsky, V. Roychoudhury, and S. Siu). New York: Kluwer, 1995.
- [12] V. Gripon, C. Berrou, *Sparse neural networks with large learning diversity*, IEEE Trans. on Neural Networks, Vol. 22, No. 7, 2011, pp. 1087-1096.
- [13] Y. Jian, H. D. Pfister, K. R. Narayanan, *Approaching capacity at high rates with iterative hard-decision decoding*, Int. Symp. Inf. Theory (ISIT), 2012.
- [14] P. Dayan, L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*, MIT Press, 2004.
- [15] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, *Efficient erasure correcting codes*, IEEE Trans. on Inf. Theory, Vol. 47, No. 2, 2001, pp. 569-584.
- [16] A. H. Salavati, A. Karbasi, *Multi-Level Error-Resilient Neural Networks*, IEEE Int. Symp. Inf. Theory (ISIT 2012), 2012.