LETTER **Communicated by Vincent Gripon**

Nested-Clique Network Model of Neural Associative Memory

Asieh Abolpour Mofrad
Asieh.Mofrad@uib.no
Matthew G. Parker
Matthew.Parker@ii.uib.no
Selmer Center, Department of Informatics, University of Bergen, Bergen 5020, Norway

Clique-based neural associative memories introduced by Gripon and Berrou (GB), have been shown to have good performance, and in our previous work we improved the learning capacity and retrieval rate by local coding and precoding in the presence of partial erasures. We now take a step forward and consider nested-clique graph structures for the network. The GB model stores patterns as small cliques, and we here replace these by nested cliques. Simulation results show that the nested-clique structure enhances the clique-based model.

1 Introduction

There has been much recent activity regarding the design of sparse neural networks in general and neuro-inspired associative memories in particular. Sparse neural networks may refer to several categories. Gripon, Heusel, Löwe, and Vermet (2015) studied models of associative memories with sparse information, that is, patterns in the learning set are random strings of 0s and 1s with about $\log n$ 1s, only. We built our model on top of the clique-based model of Gripon, Berrou, and coauthors (GB model; Gripon & Berrou, 2011), where the assumption of the GB model is that for n neurons, there are c clusters of neurons with $1 \leq c \leq \log n$ and each pattern in the learning set has exactly one active neuron per cluster.[1] We have obtained very good results by applying precoding and local coding schemes on top of the GB model (Mofrad, Parker, Ferdosi, & Tadayon, 2016).

Hopfield (2008) discussed a similar cluster-based network but from a biological perspective. In this model, neurons (n) are partitioned into a number of categories (say, c)—the cluster counterpart to the GB model—with n/c possible values. A pattern activates a single neuron in each category, and like the GB model, learning the pattern is achieved by establishing edges between the activated neurons. The number of neurons in each Hopfield

[1] In graph theory, a clique defines as a complete subgraph.

Neural Computation **29, 1681–1695** (2017)
doi:10.1162/NECO_a_00964

category is much less than in GB clusters for about the same number of neurons. As an example, compare 50 categories, each with 20 neurons versus 4 clusters, each with 256 neurons.

Although the topology and learning rule are similar, the retrieval part and learning set distribution are different. In the Hopfield model, the pattern set is generated by randomly choosing a neuron in each category, according to a power law distribution ($p(n) \sim \frac{1}{n^{1/2}}$), while in the GB model, active neurons in clusters are independent and identically distributed (i.i.d.). Moreover, the Hamming distance between two patterns in the Hopfield model is defined as the number of neurons in which they differ, while in the GB model, the Hamming distance is the number of edges in which two patterns differ, which means that distance is far better for the latter case. A precise and detailed comparison of these two models might be an interesting issue to study; however, it goes beyond the framework of this letter.

Sparse patterns lead to sparse network connections, and much research has been done on sparse networks where the patterns are chosen randomly. In these models, synaptic connections exist only between neighboring neurons. Random dilution of connections in Hopfield networks has been studied in detail (see Gardner, 1989, for instance). However, cortical connectivity is better modeled by networks consisting of several modules with dense internal connections and sparse intermodular connections, so much attention has been given to the Hopfield model in the context of small-world (Bohland & Minai, 2001) and scale-free (Stauffer, Aharony, da Fontoura Costa, & Adler, 2003) models (see also Hilgetag & Goulas, 2016, for an investigation about the human brain from the network perspective). The GB model is a sparse model, but to the best of our knowledge, it does not yet fit into this model of dense internal connections and sparse intermodular connections. The nested-clique model proposed in this letter benefits from the idea of a single active neuron in the subclusters, with dense internal connections between active neurons in a cluster and sparse connections between clusters.

On top of the GB model, several extensions has been done, including further sparse organization (Aliabadi, Berrou, Gripon, & Jiang, 2014); that is, a sparse pattern is mapped to a unique neuron of a smaller set of clusters. Replacing nonoriented connections with directed ones in a way that the network can store sequential information in a tournament-based neural network (Jiang, Gripon, Berrou, & Rabbat, 2016) is another extension.[2] A double-layered structure introduced in Jiang et al. (2016) is a good combination of a tournament-based hetroassociation as the lower layer and an upper layer in the form of clique-based autoassociative similar to the sparse networks. The nested-clique model can be considered as a two-layer

[2]In graph theory, a tournament is a directed graph obtained by assigning a direction for each edge in a complete (sub)graph

network as well, where each layer is a clique-based autoassociative network and layers intertwine with each other.

Although not pursued in this letter, part of the motivation for this work is to develop neural-inspired networks that can be potentially significantly enhanced by overlaying with quantum networks that exploit quantum nonlocality and entanglement. Moreover, since the best zero-dimensional quantum codes often appear to have a nested-clique structure that is also optimally edge sparse (Danielsen & Parker, 2004), it would then be possible to overlay the results of nested-clique associative memories with quantum graph state structures so as to enhance performance by exploiting nonlocality, superposition, and entanglement.[3]

The rest of the letter is as follow. Section 2 reviews the basics and the clique-based networks introduced by Gripon and Berrou using mostly notations from Gripon and Berrou (2011). Section 3 is devoted to the nested-clique scheme and a brief review and comparison of the local coding and precoding model. Section 4 contains the simulation results and a comparison of clique-based versus nested-clique-based networks. Section 5 concludes.

2 Clique-Based Model of Associative Memory

In the design of neural networks, a neuron is a mathematical function that models a biological neuron that receives a weighted input sum and computes its output state by a nonlinear transform. The main task of neural association is to choose the graph weights w_{ij} so that the network is able to memorize M binary patterns of length k. We are mostly interested in autoassociation: retrieving a memorized pattern from its noisy version. For a neural associative memory design, we have to determine the topology of the neural network, the learning process (updating weights between neurons), and the recalling algorithm.

In the model that Gripon and Berrou introduced, by splitting the network of n neurons into c clusters of size $l = n/c$—supposing l is a power of 2 and $\kappa = \log_2(l)$—any alphabet (say, $\mathcal{A}$) with cardinality $|\mathcal{A}| = l$ can be depicted by neurons in each cluster. Each binary pattern of length $k = c\kappa$ is then assigned to a unique set of neurons or, equivalently, to a set of characters of alphabet $\mathcal{A}$:

$$m = m_1 m_2 \cdots m_c \rightarrow (f(m_1), f(m_2), \ldots, f(m_c))$$
$$\text{where} \quad f : \{0,1\}^\kappa \rightarrow \mathcal{A}.$$

[3]However, it remains to find and construct such nested-clique structures for larger graphs, an interesting problem in graph theory, that is, to what extent they exist as the number of graph nodes grows.

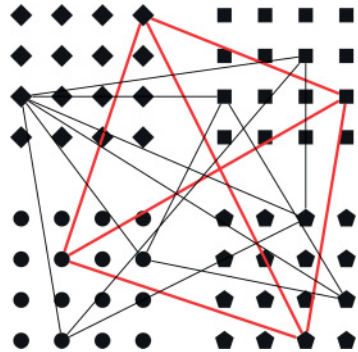


Figure 1: Learning process in a network with 64 neurons, split into four clusters of 16 neurons each. Red edges represent the binary pattern $m = (0011, 1011, 0101, 1110)$, which is learned as a clique.

The learning process is simply to connect the selected neurons together by edges to make a clique. The value of each neuron is considered binary; if a neuron is within a clique for a given pattern, its value is set to 1 and set to 0 otherwise. The weights of the edges in the network are all 1, and the learning process provides a set of edges W. This learning type (based on the Willshaw model) does not depend on the number of patterns that use neurons i and j, but on whether there is any pattern m^μ with $m_i^\mu m_j^\mu = 1$ (clipped synapses) (see Figure 1 as an example).

The recall phase or retrieving part of a partially erased learned pattern, $\hat{m}$, involves finding the most probable active neuron in each cluster and in general finding a match from the memorized patterns. Equivalently, in clique-based models, we must find a clique that contains the provided symbols as active neurons—neurons whose value is 1. Since we assume partial erasure-type error, the retrieval algorithm is slightly different from algorithms provided for GB neural networks and its variants. The detailed version of algorithm is provided in Mofrad et al. (2016) as an appendix.

The recall procedure, like general clique-based neural network models, consists of two steps: the global dynamics that establishes or eliminates connections based on provided information from erased pattern, $\hat{m}$, and a local decision that has been made to activate neurons. The winner-take-all rule activates neurons with the highest activity (or maximum degree) while the losers kicked-out rule (LsKO) eliminates active neurons with less activity using a threshold filter (see Jiang, 2014).

The modified version of the algorithm we used introduces a candidate set, $T(i)$, in each cluster, i, where partial erasure happens. The neurons in a candidate set will not be activated as long as their chances for being the final unique active neuron are equal. These candidate sets will be updated through retrieval by kicking out neurons that, based on dynamics of the

Algorithm 1: Retrieval Algorithm.

Data: $\hat{m}$, W

Result: The retrieved pattern m or failure message

begin

Based $\hat{m}$, each subpattern $\hat{m}_i$ maps to a unique active neuron n_{ij} or to a candidate set $T(i)$. /* The former occurs if $\hat{m}_i = m_i$ and the latter in the case partial erasure occurs. */

All edges from W with at least one end as an active neuron are established.

while *$\hat{m}$ not retrieved and at least a $|T(i)| = 1$ found in the last iteration* **do**

for *All $T(i)$* **do**

Find the max degree of $n_{ij} \in T(i)$: $v^i_{\max}$. The neurons with degree equal $v^i_{\max}$ remains in the set and the rest will be kicked out.

if $|T(i)| = 1$ **then** /* i.e., $T(i)$ has a unique neuron n_{ij} */

n_{ij} is activated, edges in W with one end as n_{ij} are established.

$T(i)$ is removed.

if *Cluster i with $|T(i)| \geq 2$ exists* **then** /* The active neurons provide no more information for making decisions */

Put aside clusters with active neurons /* and therefore all established edges at previous step */

Edges with at least one end in the remained candidate sets are established.

while *$\hat{m}$ not retrieved and at least a $|T(i)| = 1$ found in the last iteration* **do**

for *All $T(i)$* **do**

*nested **if** *n_{ij} does not connect to all other remaining candidate sets with at least one edge* **then**

n_{ij} is kicked out from $T(i)$. /* By kicking out n_{ij} from $T(i)$, all the connected edges are removed */

if $|T(i)| = 1$ **then**

$n_{ij} \in T(i)$ is activated, $T(i)$ is removed.

network, cannot be the final candidate for activation. It is worth mentioning that the idea behind this elimination is similar to the LsKO algorithm, but we apply the principle on candidate neurons, not active neurons, and so the elimination rule becomes different. We summarize the recall method in algorithm 1; the complete version is in Mofrad et al. (2016).

Some important parameters of a memory are diversity, capacity, and efficiency. Based on Gripon and Berrou (2011), diversity is the number of learned patterns, capacity is the maximum amount of data learned in bits,

and efficiency is the ratio between capacity and the amount of information used by the network when $M = M_{\max}$ ($M_{\max}$ is an upper bound for the number of learned patterns). Since the maximum number of edges in the clique-based model is $Q = \frac{(c-1)n^2}{2c}$ and the edges are binary, this amount of available memory ideally allows the storage of

$$M_{\max} = \frac{(c-1)n^2}{2ck} = \frac{(c-1)n^2}{2c^2 \log_2(\frac{n}{c})} \tag{2.1}$$

binary patterns, where $k = c\kappa = c\log_2(l)$, and $l = \frac{n}{c}$.

3 Extension of a Clique-Based Network to a Nested-Clique Network

Suppose the network of n neurons splits into $c_1 \times c_2$ clusters, each of size $l = \frac{n}{c_1 \times c_2}$, where l is set to be a power of 2 and $\kappa = \log_2(l)$. These clusters split into c_1 superclusters of size c_2. Each binary pattern m of length $k = c_1 \times c_2 \times \kappa$ maps to

$$\begin{aligned} m &= (m_{11}m_{12}\cdots m_{1c_2}; m_{21}m_{22}\cdots m_{2c_2}; \cdots m_{c_11}m_{c_12}\cdots m_{c_1c_2}) \\ &\longrightarrow (f(m_{11})f(m_{12})\cdots f(m_{1c_2}); f(m_{21})f(m_{22})\cdots f(m_{2c_2}); \\ &\quad \cdots f(m_{c_11})f(m_{c_12})\cdots f(m_{c_1c_2})), \end{aligned}$$

where f is again a map from subpatterns of length κ to a unique neuron

$$f : \{0,1\}^\kappa \to \mathcal{A}.$$

The learning process for pattern m is then to establish edges between active neurons in each of the c_2 clusters in c_1 superclusters; the edges will be

$$(f(m_{is}), f(m_{it})), \text{ for } 1 \le i \le c_1 \text{ and } 1 \le s \ne t \le c_2.$$

We call these edges "short connections." Similarly, active neurons in equivalent clusters of different superclusters connected with edges will be called "long connections":

$$(f(m_{is}), f(m_{js})), \text{ for } 1 \le i \ne j \le c_1 \text{ and } 1 \le s \le c_2.$$

This makes a c_1-cliques-of-c_2-cliques or $K_{c_1}[K_{c_2}]$ (see Figure 2).

Different values of n, l, c_1, and c_2 affect the memory retrieval performance, but there is no difference between a network with $c_1 = a, c_2 = b$ and learning patterns as a-cliques-of-b-cliques and a network with $c_1 = b$, $c_2 = a$ and

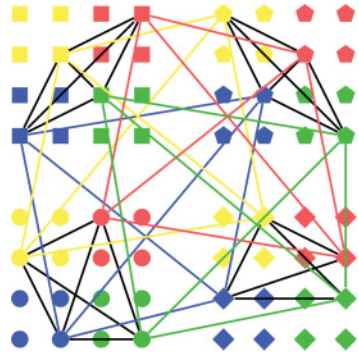


Figure 2: The learning process in a network with 64 neurons split into 4 superclusters, each with 4 clusters of 4 neurons. The 4-cliques-of-4-cliques reflect the pattern $m = (11, 01, 10, 00; 00, 10, 01, 11; 10, 00, 11, 11; 01, 11, 00, 01)$. In each supercluster, a clique of size 4 (in black) is established by short connections; activated neurons in the equivalent clusters (same colors) are connected with long connections and form a clique of size 4. Altogether the patterns are represented as a 4-cliques-of-4-cliques or $K_4[K_4]$.

learning patterns as b-cliques-of-a-cliques. It can easily be verified that after the learning process, both topologies are the same, and one can rearrange the clusters to have c_2 superclusters, each with c_1 clusters. It is equivalent to the learning pattern

$$m' = (m_{11}m_{21}\cdots m_{c_1 1}; m_{12}m_{22}\cdots m_{c_1 2}; \cdots m_{1c_2}m_{2c_2}\cdots m_{c_1 c_2})$$

instead of m, which produces exactly the same connections, but short connections become long connections and vice versa. From the information-theoretic view, there is no difference between learning a set of patterns like m' instead of the original set of patterns. However, since in general, short connections are preferable to long connections, we choose $c_2 \geq c_1$ hereafter.

Retrieval is basically the same as the clique-based version. The only difference is in the neuron removal from candidate sets, which is pointed out as ***nested** in algorithm 1. The **if** condition would be changed to

---

***nested** **if** n_{ij} *does not connect to all other remained candidate sets in the same supercluster and all other remained candidate sets in the equivalent clusters in other superclusters with at least one edge* **then**

  n_{ij} is kicked out from $T(i)$.

We can use the same argument for memory parameters when the network is a nested clique. The maximum number of possible edges is $Q' = \frac{n^2(c_1+c_2-2)}{2c_1\times c_2}$, which gives an upper bound for diversity,

$$M'_{\max} = \frac{n^2(c_1+c_2-2)}{2(c_1\times c_2)k} = \frac{n^2(c_1+c_2-2)}{2(c_1\times c_2)^2\log_2(\frac{n}{c_1\times c_2})}, \tag{3.1}$$

where $k = c_1\times c_2\times\kappa = c_1\times c_2\log_2(l)$, and $l = \frac{n}{c_1\times c_2}$.

Note that the clique-based model can be considered a special case of the nested-clique model where $c_1 = 1$ and $c_2 = c$. It can be easily seen that in this case, $Q' = Q$ and $M'_{\max} = M_{\max}$.

To compare the current work with local coding and precoding methods, which we applied in previous work (Mofrad et al., 2016), we need a brief review of those. Consider that a set of patterns of length $c\kappa$ is learned to a GB network with $c\times 2^\kappa$ neurons; c clusters of size $l = 2^\kappa$. Local coding converts each subpattern of length κ to a code word of size κ' from a chosen code with appropriate Hamming distance. The only difference with an uncoded GB is to project a code word to a neuron instead of a random subpattern. The learning process produces the same edge set W, but in retrieval, the distance between code words enhances the performance at the local level (i.e., within clusters). By the precoding technique, a set of patterns, each of length $c\kappa$, maps to a set of code words of a chosen code where code word length $\mathcal{N} > c\kappa$. Therefore, if we choose a proper code such that $\frac{\mathcal{N}}{c}$ is a natural number, then we can learn code words in a GB network of $c\times 2^{(\mathcal{N}/c)}$ neurons. As can be seen in precoding, n and l are not preserved like local coding, but for the same number of clusters, the number of edges (or Q), remains equal.

Precoding enhances local retrieval compared to the uncoded version of GB, but it is weaker than the local coding model. The main performance gain of precoding comes from generating a higher distance between cliques associated with different patterns, so precoding is suitable when a higher density is required. Both local coding and precoding manipulate the data to be learned not the network topology.

The nested-clique model is an extension of the GB model, and as we mentioned, GB can be considered a special case of the nested-clique model. We can look at the nested-clique technique in three ways. First, consider c_2 parallel GB memories with $c = c_1$ and $l = 2^\kappa$ such that c_2 individual sets of random patterns of size d are learned to them. If we consider them as superclusters and add edges to equivalent clusters (long edges), we achieve a nested-clique memory for which new edges act as the second source for association, and retrieval is enhanced in comparison to the parallel scenario.

So although we have longer patterns, the new patterns can be seen as a combination of smaller (meaningful) patterns.[4]

As the second angle, we can consider a GB network with c clusters each of size l and a given c_2, where $\frac{l}{c_2}$ is a power of 2. To achieve a nested-clique version, instead of the mapping $f : \{0, 1\}^{\kappa} \to \mathcal{A}$ where $\kappa = \log_2(l)$, a new map can be used that activates c_2 neurons in each cluster instead of just one: $f' : \{0, 1\}^{\kappa'} \to \mathcal{A}'$ where $\kappa' = \log_2(\frac{l}{c_2})$ and $|A'| = \frac{l}{c_2}$. So by adding edges between active neurons in each cluster (short connections) and connecting active neurons of equivalent clusters, the association within each cluster is higher, and therefore the retrieval rate will be enhanced.

Note that in both arguments, we add some new connections and indeed involve each subpattern in two cliques, (two constraints in coding theory terminology), which leads to higher performance. The idea of using two encodings of the same message in order to gain some benefit is not new in coding theory (see Turbo codes, for instance, in Berrou & Glavieux, 1996, and for neural networks, see Jiang et al., 2016).

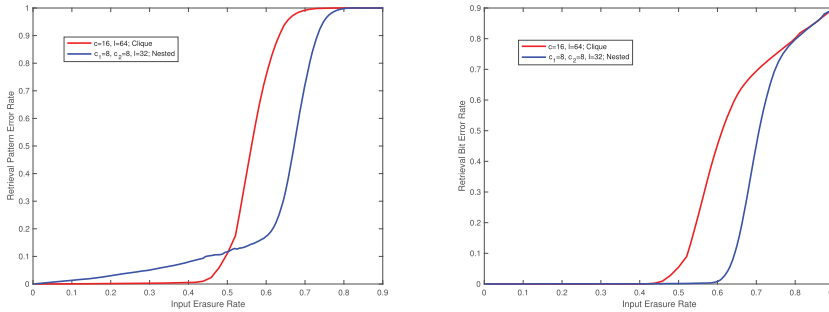
In the above arguments, we added edges to the GB network to make a nested-clique with better performance; as another possibility, one may degenerate a clique-based network to achieve a nested-clique network. Consider a GB network with $c_1 \times c_2$ clusters, so each pattern is projected to a $(c_1 \times c_2)$-clique with $\frac{(c_1 \times c_2)(c_1 \times c_2 - 1)}{2}$ connections. If we partition cliques into c_1 superclusters of c_2 cliques and take off the connections between nonequivalent clusters, we will achieve a nested-clique structure. This time, since we lose connections and therefore information, the recall performance would be weaker. Degenerated cliques are addressed in Jiang et al. (2016) where it was shown that (in Jiang's Figure 3) the retrieval performance in the presence of partial erasure decreases for degenerated cliques as expected.

From the three different ways to achieve a nested-clique network from a GB network, it can be seen that the number of connections plays a key role. To make a fair comparison of the two structures, we fixed the amount of available memory, or the number of edges, and compared nested-clique and clique-based versions in the simulation part.

4 Simulation Results

To see the retrieval performance of the proposed associative memory and compare clique-based and nested-clique-based scenarios, we fixed the number of connections or the available memory Q for both scenarios. The

[4]For example, each cluster may represent a "word," and each supercluster represents a meaningful sentence, which is a combination of c_1 words. If we have c_2 such memories and the long connections represent similar word classes (all are nouns or verbs, for example), then we have an extra knowledge, which assists in recalling the sentences in superclusters.



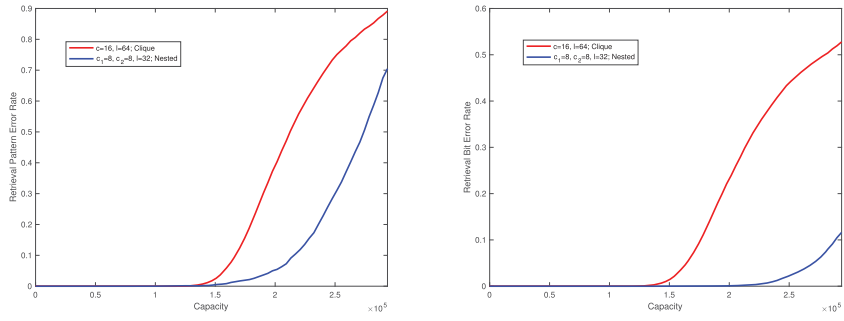
(a) Pattern Error rate (b) Bit error rate

Figure 3: Number of clusters for the red curve is $c = 16$. Each subpattern is of length $\kappa = 6$, so there are 64 neurons per cluster with pattern length 96. For the blue curve, the parameters are $c_1 = 8$, $c_2 = 8$, and $\kappa = 5$, so the number of neurons per cluster is 32 and the pattern length is 320. See Table 1 for a comparison of parameters.

symbols in the patterns are considered i.i.d. random variables. Because the data set is chosen randomly, we repeat the experiment 2500 times and compute the average to have more reliable results (i.e., we randomly choose 2500 patterns from the learning set as the input and partially erase them). We then tried retrieving the chosen pattern. If the pattern is completely retrieved, the algorithm is successful; otherwise, it fails. Both pattern error rate, which is the rate of unsuccessful retrieval, and bit error rate, which refers to the probability of one bit being erased after retrieval, are evaluated.

The first comparison is done for similar amounts of learned data (in bits) for different erasure rates, (see Figure 3), and then a comparison is made based on different amounts of learned data when the erasure rate is fixed to 0.6 (see Figure 4). The network parameters are provided in Table 1 for Figure 3 and in Table 2 for Figure 4.

In Figure 3, the number of possible edges for the GB model with $c = 16$ and $l = 64$ is 4.9×10^5. For the nested-clique structure with $c_1 = 8, c_2 = 8$, and $l = 32$, we have 4.6×10^5 possible edges, and the memory used (Q) for both is approximately the same. Since the length of patterns is longer (about three times) in nested-clique models, we choose higher diversity (about 3 times) for the GB model to have the same capacity and efficiency to compare the retrieval performance in different erasure rates. In Table 1, parameters are compared when the erasure rate is 0.6. With these parameters, we see that the retrieval pattern error rate in a nested clique is 6 times less than the GB model, and for the bit error rate, this value becomes even better—53 times less. Note that the bit error rate in the worse case equals the erasure rate while the pattern error rate in the worst case equals 1.



(a) Pattern error rate in retrieval (b) Bit error rate in retrieval

Figure 4: The number of clusters for the red curve is $c = 16$. Each subpattern is of length $\kappa = 6$, so there are 64 neurons per cluster with pattern length 96. For the blue curve, the parameters are $c_1 = 8$, $c_2 = 8$, and $\kappa = 5$, so the number of neurons per cluster is 32 and the pattern length is 320. The erasure probability is fixed at 0.6, and retrieval for different capacity is depicted. See Table 2 for a comparison of parameters.

Table 1: Comparison of Performance between the GB Model and the Nested Clique for the Same Capacity.

Model	GB	Nested Clique	Ratio
Memory used (Q)	4.9×10^5	4.6×10^5	≈ 1
Neurons (n)	1024	2048	$\times 2$
Pattern length	96	320	$\times 3$
Pattern error rate	0.79	0.13	$\div 6$
Bit error rate	0.48	0.009	$\div 53$
Diversity	2600	730	$\div 3.5$
Capacity	249,600	233,600	≈ 1
Efficiency	0.51	0.51	≈ 1

Note: In the GB model $c = 16$ and $\kappa = 6$; in the nested-clique model $c_1 = 8$, $c_2 = 8$, and $\kappa = 5$ where the erasure rate is fixed to 0.6.

In Figure 4 for the same networks and memory used (Q), we fixed the erasure rate to 0.6 and compare the two networks by the capacity factor, so diversity, capacity, and efficiency are not the same here. In Table 2 we fixed the pattern error rate to see to what extent networks can learn and recall with a pattern error rate less than 0.1. The GB model has a higher diversity 2.6 times more, but since the pattern length is smaller, the capacity in the nested clique is still better (1.3 times more than GB). As a consequence, the efficiency of the nested clique outperforms GB. So if the concern is the total information that the network is able to memorize and then retrieve in

Table 2: Comparison of Performance between the GB Model and the Nested Clique for the Same Capacity.

Model	GB	Nested Clique	Ratio
Memory used (Q)	4.9×10^5	4.6×10^5	≈ 1
Neurons (n)	1024	2048	$\times 2$
Pattern length	96	320	$\times 3$
Pattern error rate	0.102	0.1	1
Bit error rate	.06	0.002	$\div 30$
Diversity	1740	666	$\div 2.6$
Capacity	167,040	213,120	$\times 1.3$
Efficiency	0.34	0.46	$\times 1.35$

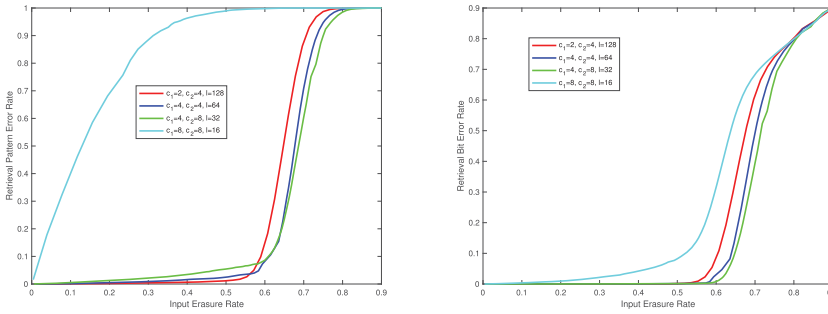
Note: In the GB model, $c = 16$ and $\kappa = 6$. In the nested-clique model, $c_1 = 8$, $c_2 = 8$, and $\kappa = 5$, where the erasure rate is fixed to 0.6 and the pattern error rate equals 0.1, which is acceptable.

the presence of partial erasure, the nested clique is better, but if diversity is the most important parameter, since GB memorizes smaller patterns, it can learn and recall more patterns within a fixed amount of memory used (Q).

Moreover, a comparison between different nested-clique scenarios for a fixed number of neurons was considered. In Figures 5 and 6 for $n = 1024$, four nested-clique configurations are compared. For the same capacity (80,640 bits) in Figure 5, we see that the nested-clique model with $c_1 = 8, c_2 = 8$, and $l = 16$ does not have good results compared to the other three configurations. The reason might be the very small size of clusters. Depending on which parameter is more favorable, one might choose a different configuration. As we can see in Table 3, the $c_1 = 2, c_2 = 4$ model has better diversity than $c_1 = 4, c_2 = 4$, which is better than $c_1 = 4, c_2 = 8$. However, the pattern retrieval rate and efficiency is in the reverse order. Since we care about capacity and the retrieval rate, the best configuration in this simulation will be $c_1 = 4, c_2 = 8$.

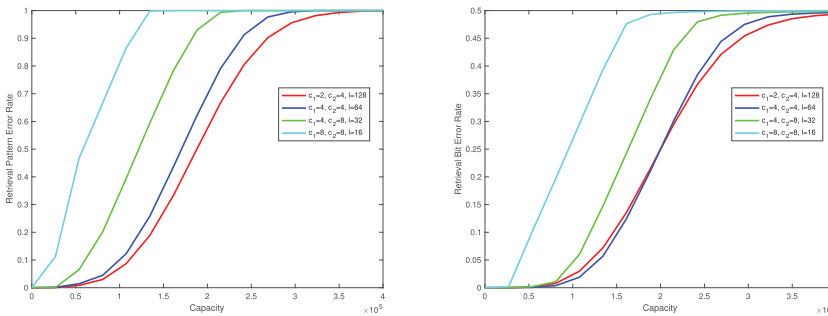
For the fixed erasure rate 0.5 in Figure 6, we see that the larger cluster size results in a better retrieval error. We fixed the acceptable retrieval error rate to 0.05 in Table 4.

The $c_1 = 2, c_2 = 4$ model has better diversity than $c_1 = 4, c_2 = 4$, but both have the same capacity. Since the number of possible connections in $c_1 = 4, c_2 = 4$ is fewer, the efficiency is better. So for the same amount of information, one can choose between better diversity or better efficiency by choosing $c_1 = 2, c_2 = 4$ or $c_1 = 4, c_2 = 4$. $c_1 = 4, c_2 = 8$ has good efficiency compared to the $c_1 = 2, c_2 = 4$ and $c_1 = 4, c_2 = 4$; however, it has less diversity and capacity. This scenario is able to learn longer patterns instead. The last scenario, $c_1 = 8, c_2 = 8$, can learn long patterns; however, its diversity, capacity, and efficiency are not comparable to the other configurations. One



(a) Pattern error rate in retrieval (b) Bit error rate in retrieval

Figure 5: A comparison of different nested-clique scenarios by changing c_1 and c_2 while the number of neurons are fixed.



(a) Pattern error rate in retrieval (b) Bit error rate in retrieval

Figure 6: A comparison of different nested-clique scenarios with a fixed number of neurons with the erasure probability set to 0.5.

might choose based on which parameter is more important; however, the $c_1 = 2, c_2 = 4$ configuration shows better capacity and diversity under the conditions of an erasure rate of 0.5 and a retrieval pattern rate of 0.05.

5 Conclusion

The nested-clique neural associative memory introduced in this letter is an extension of the clique-based (GB) mode. We have compared the retrieval capability of the two networks in the presence of partial erasures. It is possible to achieve a nested-clique network from a GB (or a set of GB) network(s) by adding extra connections, which makes the network stronger and more robust against erasure. It is also possible to remove some connections and

Table 3: Comparison between Different Nested-Clique Scenarios for Fixed Number of Neurons and Capacity When the Erasure Rate Is Fixed to 0.6.

Model	$c_1 = 2, c_2 = 4$	$c_1 = 4, c_2 = 4$	$c_1 = 4, c_2 = 8$	$c_1 = 8, c_2 = 8$
Memory used (Q)	2.6×10^5	2×10^5	1.6×10^5	1.1×10^5
Neurons (n)	1024	1024	1024	1024
Pattern length	56	96	160	256
Pattern error rate	0.129	0.0748	0.0876	0.998
Bit error rate	0.0706	0.024	0.0063	0.362
Diversity	1440	840	504	315
Capacity	80,640	80,640	80,640	80,640
Efficiency	0.31	0.41	0.5	0.73

Table 4: A Comparison of Different Nested-Clique Scenarios with a Fixed Number of Neurons Where the Erasure Rate Is Fixed to 0.5 and the Accepted Retrieval Pattern Error Rate Is Set to 0.05.

Model	$c_1 = 2, c_2 = 4$	$c_1 = 4, c_2 = 4$	$c_1 = 4, c_2 = 8$	$c_1 = 8, c_2 = 8$
Memory used (Q)	2.6×10^5	2×10^5	1.6×10^5	1.1×10^5
Neurons (n)	1024	1024	1024	1024
Pattern length	56	96	160	256
Pattern error rate	0.046	0.05	0.05	0.004
Bit error rate	0.006	0.0025	0.0012	0.0
Diversity	1920	1120	504	105
Capacity	107,520	107,520	80,640	26,880
Efficiency	0.41	0.55	0.504	0.244

degenerate a clique-based network to achieve a nested-clique network; in such a case, the nested-clique network is less robust against erasure. For a fair comparison in simulation, we fixed the number of connections and capacity and saw that the nested-clique structure outperforms the clique-based model. The nested-clique model in this case is able to learn longer patterns while its diversity (due to fixed capacity) is smaller. Both local coding and precoding techniques that improve the clique-based model can be used in the nested-clique model in the same manner if one needs to store more information and protect it from strong erasures. Local coding is especially helpful in the case that the erasure rate and diversity are both high. It is worth mentioning that one may consider other types of nested graphs—for instance, $K_{c_1}[C_{c_2}]$, which has a cycle in the lower layer. Moreover, for larger networks one expects to extend the nested-clique structure to more layers (e.g., $K_{c_1}[K_{c_2}[K_{c_3}]]$).

References

Aliabadi, B. K., Berrou, C., Gripon, V., & Jiang, X. (2014). Storing sparse messages in networks of neural cliques. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(5), 980–989.

Berrou, C., & Glavieux, A. (1996). Near optimum error correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications*, *44*(10), 1261–1271.

Bohland, J. W., & Minai, A. A. (2001). Efficient associative memory using small-world architecture. *Neurocomputing*, *38*, 489–496.

Danielsen, L. E., & Parker, M. G. (2004). Spectral orbits and peak-to-average power ratio of Boolean functions with respect to the {I, H, N} n transform. In *Proceedings of the International Conference on Sequences and Their Applications* (pp. 373–388). New York: Springer.

Gardner, E. (1989). Optimal basins of attraction in randomly sparse neural network models. *Journal of Physics A: Mathematical and General*, *22*(12), 1969–1974.

Gripon, V., & Berrou, C. (2011). Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks*, *22*(7), 1087–1096.

Gripon, V., Heusel, J., Löwe, M., & Vermet, F. (2015). *A comparative study of sparse associative memories*. arXiv:1512.08892.

Hilgetag, C. C., & Goulas, A. (2016). Is the brain really a small-world network? *Brain Structure and Function*, *221*(4), 2361–2366.

Hopfield, J. J. (2008). Searching for memories, sudoku, implicit check bits, and the iterative use of not-always-correct rapid neural computation. *Neural Computation*, *20*(5), 1119–1164.

Jiang, X. (2014). *Storing sequences in binary neural networks with high efficiency*. PhD diss., Université de Bretagne Occidentale.

Jiang, X., Gripon, V., Berrou, C., & Rabbat, M. (2016). Storing sequences in binary tournament-based neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(5), 913–925.

Mofrad, A. A., Parker, M. G., Ferdosi, Z., & Tadayon, M. H. (2016). Clique based neural associative memories with local coding and pre-coding. *Neural Computation*, *28*, 1–21.

Stauffer, D., Aharony, A., da Fontoura Costa, L., & Adler, J. (2003). Efficient Hopfield pattern recognition on a scale-free neural network. *European Physical Journal B-Condensed Matter and Complex Systems*, *32*(3), 395–399.

Received July 7, 2016; accepted January 30, 2017.