Finding All Matches in a Database using Binary Neural Networks

Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, Michel Jezequel

Finding All Matches in a Database using Binary Neural Networks

Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel and Michel Jezequel
IMT Atlantique
Brest, France
email: name.surname@telecom-bretagne.eu

***Abstract*—The most efficient architectures of associative memories are based on binary neural networks. As example, Sparse Clustered Networks (SCNs) are able to achieve almost optimal memory efficiency while providing robust indexation of pieces of information through cliques in a neural network. In the canonical formulation of the associative memory problem, the unique stored message matching a given input probe is to be retrieved. In this paper, we focus on the more general problem of finding all messages matching the given probe. We consider real datasets from which many different messages can match given probes, which cannot be done with uniformly distributed messages due to their unlikelyhood of sharing large common parts with one another. Namely, we implement a crossword dictionary containing 8-letter english words, and a chess endgame dataset using associative memories based on binary neural networks. We explain how to adapt SCNs' architecture to this challenging dataset and introduce a backtracking procedure to retrieve all completions of the given input. We stress the performance of the proposed method using different measures and discuss the importance of parameters.**

Keywords–Neural Networks, Associative Memories, Sparse Coding, Iterative Information Processing

I. Introduction

Associative memories are devices in which stored content may be addressed from part of it. Consider for instance a melody which is brought back to memory from the first music notes. Their functioning offer an alternative to classical indexed-based memories in which an absolute address is required in order to access content. For this reason, they are considered as a plausible model for human memory [1]. Associative memories are also very popular in electronics as they are key components of many systems[2]-[8]

There are basically two ways to design associative memories. In errorless applications, the content of the memory is typically indexed in such a way that it is possible to perform a fully parallel search to find a match of a given request. This is in particular the functioning of Content Addressable Memories (CAMs) [2]. When errors are tolerable, the most effective systems are based on recurrent neural networks [3]. These networks then split into two categories: binary systems and weighted ones. It is well known that weighted systems offer poorer performance than their binary counterpart [4].

Binary associative memories have been introduced in the 60s and popularized since thanks to their remarkable performance. These systems have in common that they embody pieces of information as patterns in binary graphs. For uniformly distributed messages and well scaled parameters, it has been conjectured for a long time and proven recently [5] that these systems are able to achieve very good performance asymptotically. Experiments support that performance is also very good for medium size neural networks (i.e., networks containing a few thousands of units). Because they compare favorably to other existing works [5], we decided to focus on SCNs in this paper.

A key component to obtain good performance in binary associative memories is the choice of the retrieval process. There is a vast literature on the subject [6]. However, most of the existing works focus on the scenario where there is a unique match associated with the given probe (with the noticeable exception of [7]). In this paper, we are interested in finding all stored contents that are associated with the given probe. This problem is of paramount importance when targeting applications in artificial intelligence and cognitive science [8].

In order to stress our proposed method on real datasets, we decided to focus on the implementation of a crossword dictionary able to retrieve any 8-letter english words from a partially erased input, and a chess endgame database to validate the genericity of the method. Mainly, our contributions are twofold: we show a) how to design a binary associative memory able to store then retrieve messages with almost zero error probability. For this purpose, we propose a strategy loosely based on "twin neurons" [9]. And b) we propose a backtracking solution to find all completions of the given input instead of a unique one. Our proposed solution is evaluated and parameters influence is discussed thoroughly.

The outline of the paper is as follows. In Section II, we introduce related work. In Section III, we present the proposed methodology used to store nonuniform data and to retrieve all matches associated with a given request. Mathematical analysis is performed in Section IV. Experiments results are presented in Section V. Finally, Section VI is a conclusion.

II. Related work

Solutions have been proposed in the literature in order to handle nonuniform distributions in binary associative memories. It is in particular the case in [10] for the Willshaw model. However, the lack of structure in this type of associative memories makes it difficult to propose efficient strategies.

In the context of SCNs, interesting results have been obtained using restricted models [11] inspired by the functioning of restricted Boltzmann machines. In this work, the authors propose to use a bipartite graph in which stored messages are associated with i.i.d. uniform ones. They show that this strategy allows for very efficient processing of visual signals.

A comparison of proposed approaches have been proposed in [12] and then extended and applied to real datasets in [9].

In their work, the authors show the interest of using data driven approaches in which overused parts of the network are scaled accordingly in order to counterbalance the effect of nonuniformity on performance. Our proposed solution is in the same vein.

In [13], the authors use a combination of twin neurons and a "boosting" technique in order to retrieve messages in adversarial scenarios. In [7], the authors propose for the first time to adapt retrieval procedures of neural network based associative memories in order to solve complex challenges, including finding all matches associated with a given request. In this work, the authors propose to use a simulated annealing approach to solve this specific problem, leading to very good performance at the cost of dramatically increased complexity. The solution we propose obtain exactly the same output but with reduced complexity.

III. Methodology

Our proposed solution is based on SCNs. In the next subsections, we introduce SCNs using the notations of the initial works [14][15]. We then explain how to manage nonuniformly distributed messages. Finally, we propose a solution to obtain all stored messages that are completions of the given input (instead of only one in classical SCNs).

A. Sparse Clustered Networks

Consider a finite alphabet $\mathcal{A}$ made of ℓ symbols. We call message a word over $\mathcal{A}$ containing c symbols exactly. We denote by m such a message and by m_i its i-th symbol.

SCNs are binary associative memories that are able to store a set $\mathcal{M}$ of M messages and retrieve one of them with a nonzero probability when part of its symbols are missing. More precisely, it has been shown that for i.i.d. uniform messages and for some parameters (*e.g.*, $c = \log(\ell)$, $M < 2\log(\log(\ell))\ell^2$), this probability tends to one [5].

The storage procedure is as follows: a neural network made of $c \times \ell$ units is considered. Units are split into c parts (we term them "clusters") of equal size indexed from 1 to c. Inside each cluster, units are then indexed using symbols of $\mathcal{A}$, *i.e.*, each unit is associated one-to-one with a symbol in $\mathcal{A}$. As a result, each unit is uniquely determined by a couple (i, a), where $1 \leq i \leq c$ and $a \in \mathcal{A}$.

It is thus possible to associate a message with a set of c units, through the function:

$$f : m \mapsto \{(i, m_i), 1 \leq i \leq c\}.$$

To store the messages contained in $\mathcal{M}$, the procedure consists in, starting from a neural network with no connection, adding all connections between units in $f(m)$ for each message $m \in \mathcal{M}$. Let us denote by $W_{(i,a)(i',a')}$ the adjacency matrix of the neural network ($W_{(i,a)(i',a')} = 1$ iff units (i, a) and (i', a') are connected). This process is illustrated in Figure 1. In Figure 1 the alphabet is $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ and $c = 4$. The stored messages are $m_1 = [a_1, a_1, a_3, a_2]$ and $m_2 = [a_1, a_4, a_4, a_3]$. Connections added by the storage of m_1 and m_2 have been depicted differently (dashed vs. dotted lines) in order to ease reading, but connections in the network are binary and thus not labelled.

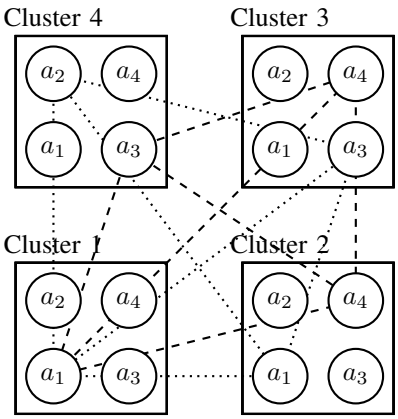

Figure 1. Illustration of the storage procedure in SCNs.

Once a set of messages has been stored, an iterative retrieval procedure is used to recall one of them from a partially erased probe.

Consider a message $m \in \mathcal{M}$, we introduce the erasure function:

$$\tilde{}: m \mapsto \tilde{m} \text{ such that } \forall i, \tilde{m}_i = m_i \vee \tilde{m}_i = \perp,$$

where $\perp \notin \mathcal{A}$ denotes an erased symbol. Consider an indicator function $v : \{1, 2, \ldots, n\} \times \mathcal{A} \to \{0, 1\}$ which associates a unit with its binary activation state (active or not).

Considering $\tilde{m}$ as an input, an optimized retrieval procedure [6] consists in repeating the following two steps, starting with v^0 the indicator function of $f(\tilde{m})$:

1) Estimate a likelihood score for each unit in the network to be activated, based on the connection they share with other units in the network. To do so, for each unit (i, a) is computed the score $s_{i,a}^{t+1} = \sum_{i'=1}^{c} \max_{a' \in \mathcal{A}} \left[W_{(i,a)(i',a')} v_{i',a'}^t \right]$. In other words, for each unit we count how many clusters of the neural network contain an activated unit which they are connected to.
2) Based on the previously computed score, select the units to activate or not. Here, we simply activate the units with the maximum score among their clusters.

It can easily be shown that this iterative procedure converges as the set of activated units is nonincreasing with iterations, starting at the second iteration [5].

The converged state v^∞ corresponds to the output of the neural network. In case a unique unit is activated in each cluster, this state can be mapped to the corresponding message m such that v^∞ is the indicator function of $f(m)$.

Obtaining mathematical proofs of performance can prove to be challenging [5]. This is why many works make the simplifying assumption that, when messages to store are i.i.d. uniform, existence of connections in the neural network can be considered independent. Numerous experimental works justify this assumption [15]. In this context, it is possible to derive probability of success in retrieving a stored message from a

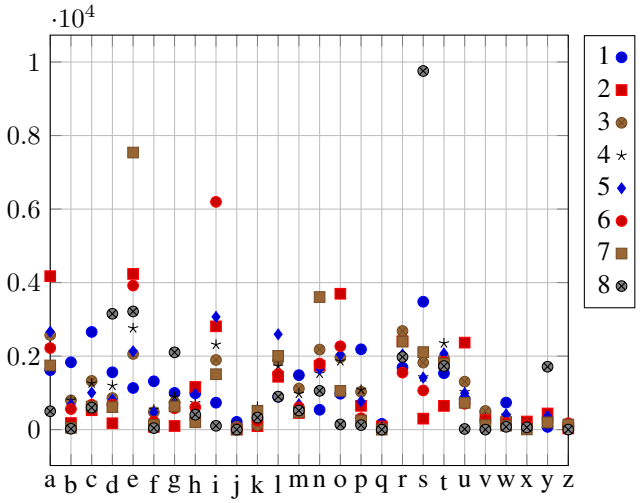

Figure 2. Histogram representing the frequency of apparition of each symbol ('A' to 'Z') for each possible position (1 to 8).

partially erased probe containing c_e erased symbols [15]:

$$P_e = 1 - \left(1 - \left(1 - \left(1 - \ell^{-2}\right)^M\right)^{c-c_e}\right)^{(\ell-1)c_e}.$$

B. Nonuniformity of Stored Messages

It is well known that nonuniformity of stored messages can lead to dramatic loss in performance [10]. Many solutions have been proposed [12]. All of them consist in adding material, either units in each cluster or new clusters, in order to counterbalance the effect of nonuniformity.

The technique known to offer best performance is called "twin neurons" [9]. It consists in duplicating overconnected units in the network. The method we propose in this paper, is inspired by this mechanism.

First, let us introduce the first dataset used in this paper. We decided to use english words made of 8 letters. We use a database containing $M = 28'557$ such words [16]. Obviously this database contains a nonuniform distribution of words, such that some letters are more frequent than others. As an illustration, Figure 2 depicts the frequency of apparition of each symbol at each possible position. Using the classical method – namely we assign the same number of units with each symbol – some units are saturated with connections, leading to dramatically low performance. To avoid this, we duplicate units in the neural network corresponding to the overused symbols, such that the connections are divided accordingly. This process is depicted in Figure 3 and more precisely described in the following paragraphs.

In our proposed method, words of 8 symbols are represented using 8 clusters (one per position in the word). But contrary to classical SCNs, we use more than the required 26 units. Specifically, in order to represent a symbol at a given position, we use possibly more than a single unit. Consider symbol a at position i, and denote λ (here λ is no longer the cardinality of $\mathcal{A}$) the number of units in each cluster, $W(i,a)$ the number of eight symbol which i-th one is a, then the number of units representing a in cluster i is

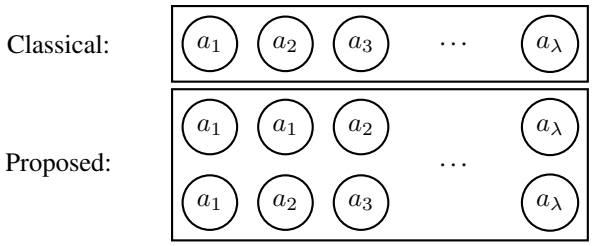

Figure 3. Illustration of the proposed solution to duplicate units corresponding to more frequent symbols.

$$N(i,a) = \frac{W(i,a)}{M}\lambda.$$

Said otherwise, units representing symbol a at position i are proportional to the frequency of 8-letter words containing symbol a at position i.

The storing process is then modified. As a matter of fact, there is now multiple choices of units to activate in each cluster. The idea is to choose one of them alternatively in order to balance the number of connections per unit. This is depicted in Figure 4. Depending on the frequency of each symbol at each position, some units have been duplicated. The first stored message is $[a_1, a_1, a_3, a_3]$, as depicted in the first step. Because there are two units representing symbol a_1 in cluster 1, an arbitrary choice has been performed. The second message to store is $[a_1, a_2, a_2, a_1]$, making use again of symbol a_1 in cluster 1. This time the other unit has been chosen to balance connections in the neural network. The first added message makes use of a unit representing symbol a_1 in cluster 1 and the next message makes use of another unit representing a_1 in cluster 1.

C. Finding all Matches of a Given Request

Crosswords players are familiar with configurations where a few characters erased can lead to many different combinations. This is typically not expected with uniformly distributed messages, as the probability λ^{-c_0} two words share c_0 common given symbols vanishes exponentially fast to zero with c_0.

When multiple stored messages are completions of the input probe, the retrieval process is expected to converge to a state where active units contain at least the union of the units corresponding to the different possible outputs. Finding which unit is associated with each other is a combinatorial problem that may prove challenging in practice.

To illustrate the problem, consider that at the end of the retrieval process, 10 units are activated in the 4 initially erased clusters. A possible explanation is that there are 10 completions of the initial probe, corresponding to the activation of its own unit in each cluster. But, in terms of combinatorial possibilities, there are $10'000$ possibilities.

In practice, having to check every single possibility would lead to considerable increase in complexity. This is why we introduce a backtracking alternative solution.

Suppose we have a set of activated units s_i in the i-th cluster. We propose to select one of them arbitrarily, unactivate all others, and pursue the retrieval process. Once

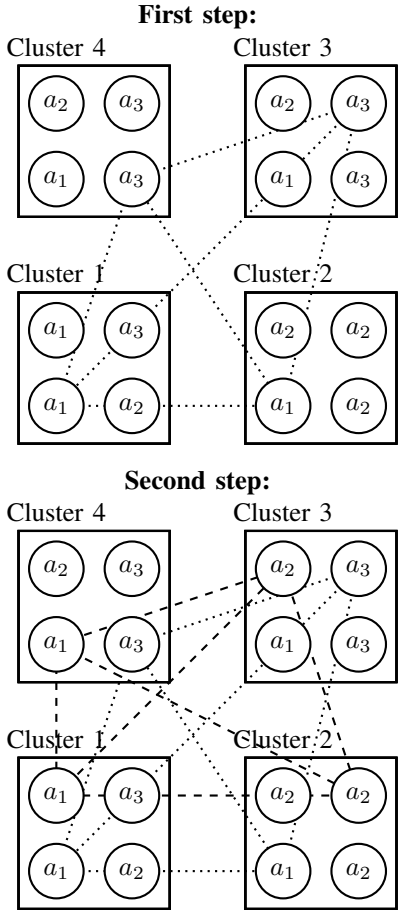

Figure 4. Illustration of the updated storage procedure. Here the alphabet is $\mathcal{A} = \{a_1, a_2, a_3\}$ and $c = 4$.

it is completed, we unactivate the previously selected unit and remove it from s_i, and select another unit arbitrarily in s_i to start the process again. Once all units have been activated once in s_i, the process is over. Such strategy is commonly known as backtracking in computer science literature.

In ideal conditions, that is to say when solutions are nonoverlapping over the initially erased clusters, this procedure reaches a complexity that is linear with the number of solutions. This is why it is expected to perform well in practice. On the other hand, the complexity is upper bounded by the product of cardinalities of s_i, which correspond to the combinatorial factor previously introduced.

This backtracking solution offers another advantage when combined with the twin neurons strategy described in the previous subsection. As a matter of fact, selecting arbitrarily a unit in each cluster when storing a message is a good strategy to balance connections, but as a result this association is lost and when a probe is given to the network, all units corresponding to the given symbols should be activated. Said otherwise, a lot of spurious units are likely to be activated at the beginning of the retrieval process, due to the duplication of some units in the neural network.

In order to avoid this added difficulty, we propose to use the same backtracking solution previously described at the beginning of the process also, in order to erase most of spurious units at the beginning of the retrieval process.

We can illustrate this by a good example from the database. For $\lambda = 512$, consider the words aardvark and aardwolf, that could be addressed from the request aard****, where '*' denotes an erased symbol. For this scenario, we obtain 29 units corresponding to the letter 'a' in the first cluster, 75 to the letter 'a' in the second cluster, 48 to the letter 'r' in the third cluster and 21 to the letter 'd' in the fourth cluster. Thus, there are $2'192'400$ possible combinations. Using the backtracking algorithm to reduce this number, we obtain only 2 units active in each of these clusters. Actually, these two active units in each cluster are those corresponding to the two words aardvark and aardwolf exactly.

IV. Mathematical Analysis

The overall procedure corresponding to the retrieval of the messages matching a given input probe is summarized in Algorithm 1.

1. Activate all units corresponding to the nonerased symbols of $\tilde{m}$
2. Use the backtracking algorithm to remove some of the spurious units
3. Perform the decoding procedure
4. Use the backtracking algorithm to obtain guesses

Algorithm 1: Algorithm used to retrieve the stored messages corresponding to an initially partially erased probe $\tilde{m}$.

A first result is that Algorithm 1 will output all of the messages in $\mathcal{M}$ that match the probe $\tilde{m}$:

Proposition 1. *Consider a binary associative memory in which the messages in $\mathcal{M}$ have been stored. For any message $m \in \mathcal{M}$, the output of Algorithm 1 given $\tilde{m}$ as input contains all the messages in $\mathcal{M}$ that are completions of $\tilde{m}$.*

Proof: The proof is a straightforward adaptation of the proof for the classical SCNs in [5]. It is mainly based on the fact a stored message which is a completion of the input will always achieve the maximum scores in the retrieval process, and therefore the corresponding units will remain activated. ■

Note that this result does not imply that the output messages given by Algorithm 1 is exactly the correct answer, but only that it contains the correct answers. In practice, it is expected that for some queries the output contains more messages than it should.

Another interesting result is that, for large enough values of λ, the obtained associative memory has vanishing error probability:

Proposition 2. *Consider a set of messages $\mathcal{M}$. Then, the probability the output messages given by Algorithm 1 for some query $\tilde{m}$ with $m \in \mathcal{M}$ is exactly the set of messages in $\mathcal{M}$ that are completions of $\tilde{m}$ tends to one as λ tends to infinity.*

Proof: Indeed, consider the extreme case where messages in $\mathcal{M}$ are stored in the binary neural network using completely disjoint sets of units. Using very simple arguments of binomials, this happens with probability that tends to one as λ

tends to infinity. In such situation, step 2 of Algorithm 1 will keep active only the units that correspond to the targeted set of messages. Indeed, a unit can only remain active if sharing connections with all initially active units, which happens by definition only if part of a targeted message. For similar reasons, a message retrieved after step 4 of Algorithm 1 is such that all its units are interconnected, so it must correspond to a stored message. Because it is in particular connected with all the initially activated units, it is a targeted message. ■

Obviously, in practical applications, it is of paramount importance to find a good tradeoff between precision and complexity of the method, varying the value of λ. This will be discussed in the next section.

V. Experiments

In this section, we present an analysis of complexity and error rates of the proposed method, in comparison with the one described in [7]. We refer to complexity as the average number of elementary operations, which are arithmetical operations or memory accesses.

First, we examine the influence of cluster size λ on complexity using the 8-letter words dataset. The number of elementaru operations is 409 millions for $\lambda = 256$, 64 millions for $\lambda = 384$ and 8.1 millions for $\lambda = 512$.

Interestingly, larger networks have dramatically lower complexity than small networks. This phenomenon can be easily explained by examining the influence of network size on error rates results; as error decreases with increasing network size, the size of the search space decreases because fewer units are interconnected. As a consequence, the proposed backtracking algorithm eliminates more cases, resulting in a reduction of complexity.

We then compute the error rate of the proposed method. We note N the number of examples in the dataset. The add result error rate (ARER) is defined using the number P of undesirable cases obtained (cases which do not figure on the dataset). The forget result error rate (FRER) is defined using the number F of cases which figure in the dataset, but are not obtained by the method. These two error rates can be expressed in the following way:

$$ARER = \frac{P}{P+N} \qquad FRER = \frac{F}{N}$$

As stated in Proposition 1, the proposed method will always output at least the actual matches in the dataset, so FRER is equal to zero. Unless stated otherwise, we will refer to the ARER when using the term ”error rate”.

We examine error rate as a function of a) the number of stored messages and b) the size of the network, expressed as the number of units λ in each cluster. To do so, we randomly select a message $m \in \mathcal{M}$, and then erase randomly 4 symbols, getting a partially erased probe $\tilde{m}$ as an input of Algorithm 1. We repeat this procedure to compute the average error over the testset. Figure 5 shows simulation results for the 8-letter words. A network containing 256 units cannot memorise more than 7100 Messages, otherwise it retrieves them when they are half erased with a low probability (Figure 5), so it cannot handle the entire testset. A network with 384 units obtains a 3% error rate, while a network with 512 units retrieves half

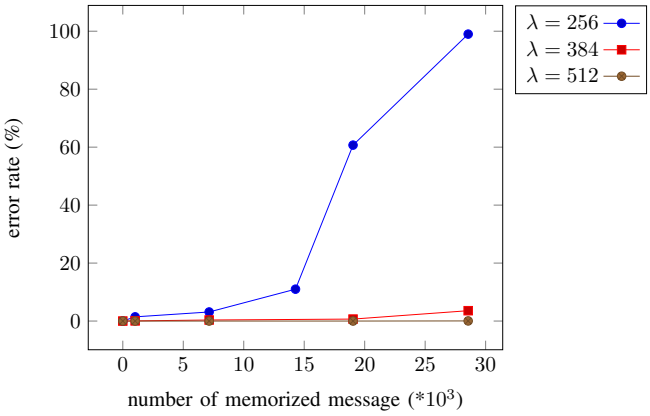

Figure 5. Error rate as a function of the number of stored messages and network size λ for the 8-letter words dataset. In these experiments, the retrieval procedure performs 4 iterations.

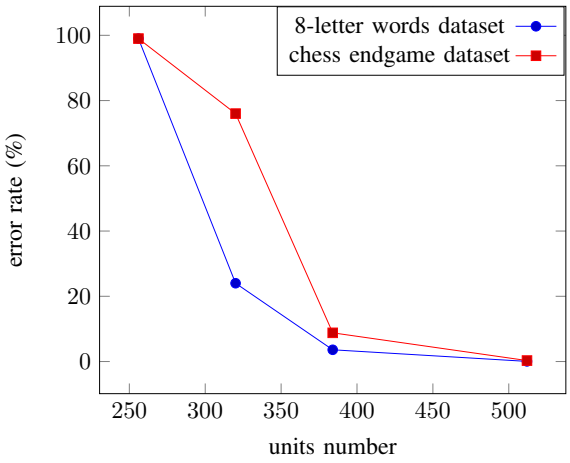

Figure 6. Error rate as a function of network size λ in two different datasets. Retrieval using 4 iterations.

erased messages with an error rate approaching the zero (blue line in Figure 6).

We use a second dataset to validate the genericity of the method. The chess endgame database [17] contains 7 attributes which represent white King position (2 attributes), white Rook position (2 attributes), black King position (2 attributes) and the optimal depth-of-win for white (1 attribute). The test protocol is the same as for the 8-letter words dataset, except that we randomly erase 3 symbols instead of 4. Figure 6 depicts for both datasets a decrease in error rate when increasing the number of units in the cluster.

Importantly, we observe that a network with 512 units leads to an optimal solution, in the sense that it minimizes both error and complexity. We choose this network to perform a comparison with the work in [7] which featured a network with 8 clusters of 512 units. Table I compares the two methods in terms of complexity and the two types of error rates.

The complexity of the proposed method is largely reduced. The ARER is almost the same for both methods, while FRER is 43% for [7] and 0% for the proposed method. This demonstrates that the proposed method leads to substantial enhancements both in terms of complexity and error rates

TABLE I. Comparing the complexity and the errors of our method and the method described in [14]

	Elementary operations	ARER	FRER
Proposed solution ($\lambda = 512$)	8.1 millions	0.045%	0%
Method described in [14]	878 millions	0.05%	43%

demonstrates that the proposed method leads to substantial enhancements both in terms of complexity and error rates compared to previous work.

VI. Conclusion and Future Work

We introduced a method to store nonuniform messages in networks of neural cliques and a method to find all matches associated with a given query. The proposed method involves the use of additional computational resources, but offers asymptotically ideal precision.

We stressed the efficiency of our method on two challenging datasets, an 8-letters english words dataset and a chess endgame dataset. We demonstrated the ability of our solution to obtain very good precision while keeping computational complexity largely reduced compared to previous work.

In future work, we will leverage the full potential of this method by developing parallel hardware architectures derived from the proposed algorithm. In addition, we plan to develop methods to automatically select hyperparameters (in particular the value of λ).

References

[1] J. R. Anderson and G. H. Bower, Human associative memory. Psychology press, 2014.

[2] T. Manhas and M. Wang, "Scalable packet classification using associative memory." Google Patents, 2012.

[3] H. Jarollahi, N. Onizawa, T. Hanyu, and W. J. Gross, "Associative memories based on multiple-valued sparse clustered networks," in 2014 IEEE 44th International Symposium on Multiple-Valued Logic. IEEE, 2014, pp. 208–213.

[4] S. Di Carlo, P. Prinetto, and A. Savino, "Software-based self-test of set-associative cache memories," vol. 60, no. 7. IEEE, 2011, pp. 1030–1044.

[5] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Improving the accuracy of network intrusion detection systems under load using selective packet discarding," in Proceedings of the Third European Workshop on System Security, ser. EUROSEC '10, 2010, pp. 15–21.

[6] E. Lehtonen, J. H. Poikonen, M. Laiho, and P. Kanerva, "Large-scale memristive associative memories," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 3, 2014, pp. 562–574.

[7] R. S. Indeck, R. K. Cytron, and M. A. Franklin, "Associative database scanning and information retrieval," May 31 2011, uS Patent 7,953,743.

[8] H. J. et al, "A non-volatile associative memory-based context-driven search engine using 90 nm CMOS MTJ-Hybrid logic-in-memory architecture," Journal on Emerging and Selected Topics in Circuits and Systems, vol. 4, 2014, pp. 460–474.

[9] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE Journal of Solid-State Circuits, vol. 41, no. 3, 2006, pp. 712–727.

[10] V. Gripon and M. Rabbat, "Maximum likelihood associative memories," in Proceedings of Information Theory Workshop, 2013, pp. 1–5.

[11] G. Palm, "Neural associative memories and sparse coding," Neural Networks, vol. 37, 2013, pp. 165–171.

[12] V. Gripon, J. Heusel, M. Lwe, and F. Vermet, "A comparative study of sparse associative memories," Journal of Statistical Physics, vol. 164, 2016, pp. 105–129.

[13] A. Aboudib, V. Gripon, and X. Jiang, "A study of retrieval algorithms of sparse messages in networks of neural cliques," in Proceedings of Cognitive 2014, 2014, pp. 140–146.

[14] A. Aboudib, V. Gripon, and B. Tessiau, "Implementing relational-algebraic operators for improving cognitive abilities in networks of neural cliques," in Proceedings of Cognitive, 2015, pp. 36–41.

[15] T. Kohonen, Self-organization and associative memory. Springer Science & Business Media, 2012, vol. 8.

[16] B. Boguslawski, V. Gripon, F. Seguin, and F. Heitzmann, "Twin neurons for efficient real-world data distribution in networks of neural cliques. applications in power management in electronic circuits," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 2, 2016, pp. 375–387.

[17] A. Knoblauch, G. Palm, and F. T. Sommer, "Memory capacities for synaptic and structural plasticity," Neural Computation, vol. 22, no. 2, 2010, pp. 289–341.

[18] R. Danilo, V. Gripon, P. Coussy, L. Conde-Canencia, and W. J. Gross, "Restricted clustered neural network for storing real data," in proceedings of GLSVLSI conference, 2015, pp. 205–210.

[19] B. Boguslawski, V. Gripon, F. Seguin, and F. Heitzmann, "Huffman coding for storing non-uniformly distributed messages in networks of neural cliques," in proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, volume 1, 2014, pp. 262–268.

[20] X. Jiang, M. R. S. Marques, P.-J. Kirsch, and C. Berrou, "Improved retrieval for challenging scenarios in clique-based neural networks," in International Work-Conference on Artificial Neural Networks. Springer, 2015, pp. 400–414.

[21] V. Gripon and C. Berrou, "A simple and efficient way to store many messages using neural cliques," in Proceedings of IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain, 2011, pp. 54–58.

[22] ——, "Sparse neural networks with large learning diversity," IEEE transactions on neural networks, vol. 22, no. 7, 2011, pp. 1087–1096.

[23] "Eight letters words database," http://www.morewords.com/wordsbylength, accessed: 2016-10-03.

[24] "Chess database," http://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King%29, accessed: 2016-10-03.