# Clone-Based Encoded Neural Networks to Design Efficient Associative Memories

Hugues Wouafo, Cyrille Chavet, Philippe Coussy

# Clone-based Encoded Neural Networks to Design Efficient Associative Memories

Hugues Wouafo, *Student Member, IEEE*, Cyrille Chavet, *Senior Member, IEEE* and Philippe Coussy, *Senior Member, IEEE*
*Lab-STICC (UMR CNRS 6285)*

***Abstract*—In this paper, we introduce a neural network model named Clone based Neural Network (CbNN) to design associative memories. Neurons in CbNN can be cloned statically or dynamically which allows to increase the number of data that can be stored and retrieved. Thanks to their plasticity, CbNN can handle correlated information more robustly than existing models and thus provides better memory capacity. We experiment this model in Encoded Neural Networks also known as *Gripon-Berrou* neural networks. Numerical simulations demonstrate that memory and recall abilities of CbNN outperform state of art for the same memory footprint.**

***Index Terms*— Associative Memories, Neural Networks, Content Addressable Memories.**

## I. Introduction

The human brain is a powerful machine able to realize complex operations like abstracting, memorizing and reasoning. It processes information through a complex hierarchical associative memory organization distributed across complex neural network (NN). These memorizing and processing capabilities associated with very high power efficiency of the brain are major features which scientists try to understand and mimic. Numerous computational models have been explored, providing the promise of practical applications in many domains (computer sciences, electronics…). Several neural networks have been proposed that are able to first memorize associations between data, and then retrieve a given data (or its associated label) when receiving an altered version of it. Systems with such capabilities are referred as associative memories [1][2]. Hopfield neural networks (HNNs)[3], Boltzmann machines [6], Kohonen maps [4], Willshaw networks [6] or Cogent Confabulation models [7] are models classically used to design such memories. Many studies have been performed to enhance their *capacity* (i.e., the maximal amount of data that can be stored and then retrieved).

In this paper, we introduce a neural network model named Clone based Neural Network (CbNN) to design associative memories. The main advantage of CbNNs is that they can handle correlated information more robustly than other existing models do. This is achieved by *cloning* neurons to increase the quantity of data that can be stored. In other words, the neural network is customized to adapt itself to the data distribution.

In this work, we experiment CbNNs in Encoded Neural Networks (ENN), also known as Gripon-Berrou neural network [8]. This ENN model is interesting from theoretical point of view, since it is a kind of "clustered" version of Willshaw, that offers much more *capacity*, and moreover it is closer to real biological neuron networks.

This paper is organized as follows. Section II presents existing models and architectures for associative memories. Section III, introduces the ENN model and describes its evolutions. Section IV formally defines the concept of CbNN and details the different steps for the storage and retrieval processes. Section V and Section VI describe the different variants for the main classes of CbNNs while comparing their performances (storage capacity, recall ability). Finally, Section VII compares the best static and dynamic CbNN variants.

## II. Related works

Two main neural network topologies are traditionally proposed in the literature: either the neurons are connected through multiple layers in a directed graph (i.e., feed-forward models [9]) or the connections between neurons form a cycle (i.e., recurrent models [3][4]).

Deep learning has brought back feed-forward networks in the spotlight through classification applications. However only few considerations from this domain are relevant since hereby we focus on recurrent models to design associative memories. Hence in [10], the authors show how to combine several networks to improve the number of retrievings. The basic idea is to select the final retrieved data once a consensus is obtained (i.e., once each copy of the network has voted). Each copy is trained to be a predictor and different combinations of learning parameters (i.e., synaptic weights) are considered. This idea of relying on a set of elements to design associative memory is one of the core principle of our work. However, instead of using different learning method parameters in each network, we propose to clone neurons or sub-networks. In [11] the authors propose to use a learning step based on a meta-heuristic ("annealed synaptic dilution") under a limited number of synapses to design associative memory. The proposed approach considers fitness and degree of neurons to determine the probability to add new connections. In our work, we propose a competition between clones of neurons which is based on a heuristic to compute a fitness metric. In [12], the authors propose an algorithm for deep learning models called Elastic Weight Consolidation (EWC). By mimicking the mammalian brain inner work, this algorithm tries to strengthen the synaptic weights during the learning step. The proposed model has been

This work has received a French government support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01.

H. Wouafo, is with Université Bretagne Sud, Lorient, France. He was Post-Doc in Lab-STICC laboratory, France.
C. Chavet, is with Université Bretagne Sud, Lorient, France. He is associate professor in Lab-STICC laboratory, France.
P. Coussy, is with Université Bretagne Sud, Lorient, France. He is full professor in Lab-STICC laboratory, France.

designed in order to preserve, or more precisely to adjust non-binary weights. As it will be exposed later in this paper, this weight preservation proposed by EWC is an inherent concept of our binary weighted model, but since we only consider binary values there is no need for weight consolidation. In [13], a machine learning model referred as Differentiable Neural Computer (DNC) is introduced. This model splits memory and computation in two separate entities, with a unique memory interface to refresh concurrently all the synaptic weights once the learning step is done. However, these memory accesses (read or write) add significant overhead to the overall performance of the system.

Recurrent models which are traditionally used to design associative memories are less addressed in the recent literature. The underlying principle is to create internal states allowing to exhibit dynamic behavior while retrieving stored messages from a part of it. Hopfield networks [3] have been defined such that their dynamic convergence is guaranteed. However, their *efficiency* (i.e., the number of stored data divided by the total number of data to store) tends to zero when the amount of stored data grows; this is even true when considering scale free topology [14][15][16]. In [13], the authors introduce an architecture based on an Associative Memory and Recall (AMR) model. The idea is to take advantage of several Hopfield-inspired networks gathered in a recursive and recurrent topology. Some of the underlying concepts of this publication could be similar to our cloning approach. However, if this proof-of-concept is interesting, the paper does not expose enough information to perform fair comparisons. Recently, binary cluster-based neural networks named *Encoded Neural Networks ENNs* have been proposed [8]. The approach was motivated both by biological considerations and ideas from information theory. This model offers a greater or equal *capacity* than classical recurrent models. ENNs use binary weights, and their neurons are gathered into groups, named *clusters*, such that a neuron in a given cluster can only be connected to neurons of the other clusters. Thanks to their binary weights and to a simple retrieval process, ENNs allow to design energy efficient Content Addressable Memories (CAM)[19]. However, as shown in [20] they only achieve an optimal performance when stored data are uniformly and identically distributed. Unfortunately, in practice, data mostly have non-uniform distribution which hence degrades ENN efficiency severely. Several approaches have been proposed to overcome these limitations at the cost of an important increase in terms of complexity and memory cost [20] (see section III for details).

In [10], we introduced a *Clone-based Neural Network* (*CbNN*) model. Cloning consists in replicating the neurons in the network in order to improve ENN results: several entities named *clones* may represent the same neuron. Using clones does not only allow to increase the number of data that a network can store and retrieve, but it also allows CbNNs to adapt to the data distribution and thus to compete with [20]. The concept of clones has been reused in [22] through neuron duplication to improve the work from [20]. However, in [10] our work was restrained to static allocation of clones.

In this paper, we target two major improvements compared to the work introduced in [10]: the network can be itself cloned (cloned networks are referred as *sub-networks*), and the clones can be dynamically allocated. In other words, during the storing step, the CbNN adapts itself in terms of number of sub-networks and number of clones, depending on the input data values to minimize the risk of *retrieval error*. The network/neuron cloning combination, which can be performed statically or dynamically, results in a large variety of models. To clearly introduce these different models, an extensive taxonomy of the CbNN model in the ENN context is proposed and variants are compared in terms of storage capacity and recall ability. Our experiments show that the best variants outperform the state-of-the-art models, for the same memory cost or widely reduce the memory cost for a given efficiency level. Before exploring these models, we first remind the original ENN basics and analyze its strengths and weaknesses.

## III. Encoded Neural Networks: Principles and Limitations

### A. Encoded Neural Network basics

An ENN [8] is based on binary sparse model. The main idea is to divide up sets of neurons into disjoint *clusters*. A depicted in Fig. 1, a cluster contains a subset of neurons from the network. Neurons that belong to the same cluster cannot be connected to each other. However, any neuron of a given cluster can be connected to any neuron in any other cluster. More precisely, the network consists of $N$ binary neurons arranged into $C$ equally-partitioned clusters. Therefore, each cluster contains $L = N/C$ neurons. An ENN with such configuration is noted $ENN(C, L)$.

In this model data are called *messages*. An ENN(C, L) can store and retrieve a message $m$ with $C$ symbols, each symbol containing $B = \log_2(L)$ bits. Considering such message, each symbol is associated with a specific cluster in the network. The symbol value serves to trigger the activation of a specific neuron in the cluster. Therefore, each neuron represents a specific symbol value and a neuron is activated by setting its *state* to 1 (when the value it is associated with is presented to the network).

Initially ENNs do not have any stored information (i.e., all the weights in the network are set to 0). During the "*storage*" step, the network memorizes all the input messages. To store a message, the activated neurons (i.e. the set of neurons corresponding to the values of the input message symbols) are connected to each other and form a *neural clique* [23] (thanks to the use of binary weighted connections). To memorize a connection between one neuron $i$ in the cluster $j$, $n_{(i,j)}$ and one neuron $i'$ in the cluster $j'$, $n_{(i',j')}$, each active neuron locally stores the value 1 in its corresponding synaptic weight $w$ (i.e. $w(n_{(i,j)}; n_{(i',j')}) = w(n_{(i',j')}; n_{(i,j)}) = 1$). Once all the messages stored, this associative memory can be used to retrieve partially erased messages (see Fig. 1 for a pedagogical example).

The retrieval process, also named *decoding* step, recovers a message when some of its symbols are unknown (because of an *erasure*). This iterative process is based on two steps: a *Scoring* step, followed by a *Local Winner Takes All* (L-WtA) step. This process is able to detect which neuron, in a cluster associated

with a missing part of the message, is the most *stimulated* one [24] (i.e., the neuron that is most likely the one missing in the erased input message).

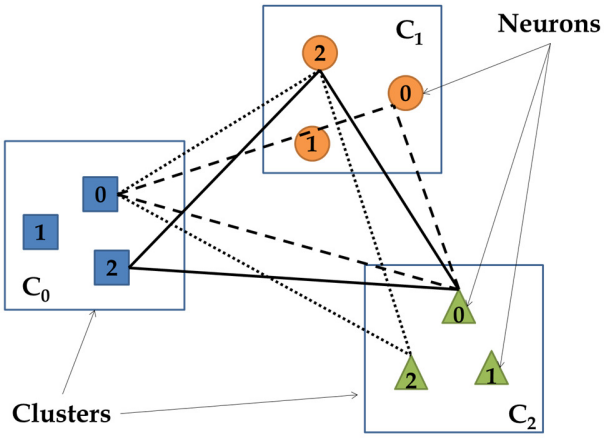



Fig. 1 An example of an ENN(3,3) with three stored messages (cliques): [0,0,0], [0,2,2] and [2,2,0]. Nota bene [0,2,0] forms a clique even if the corresponding message has not been explicitly stored.

Equation (1) defines the scoring function used to compute the *score* $sc^{t+1}(n_{(i,j)})$ of a neuron $n_{(i,j)}$at time $t+1$. This score depends on two parameters: (1) the state values of all the neurons $i'$ from all the other clusters $j'$ (i.e. $n_{(i',j')}$) computed at the time instant $t$ (i.e., in the previous iteration of the decoding) and (2) the corresponding synaptic weights (i.e., value of $w(n_{(i,j)};n_{(i',j')})$) that have been stored during the storage process (see Eq. (2)).

$$\forall\, i \in [0..L-1], j \in [0..C-1],\quad sc^{t+1}(n_{(i,j)}) = v^{t}\,(n_{(i,j)}) + \sum_{j'=0,\mathrm{j}\neq\mathrm{j}'}^{C-1} \max_{0\le k\le L-1}\left(v^{t}(n_{(i',j')}) * w(n_{(i,j)};n_{(i',j')})\right) \quad (1)$$

$$v^{t+1}(n_{(i,j)}) = \begin{cases} 1 & if\ sc^{t+1}(n_{(i,j)}) = \max\limits_{0\le i<L}\left(sc^{t+1}(n_{(i,j)})\right) \\ 0 & otherwise \end{cases} \quad (2)$$

The decoding is successful when at the end of the process there is only one active neuron in each cluster. A successful decoding may require a few iterations. When the decoding is over, the clusters which originally had no selected neuron are provided with the selection of a neuron or a group of neurons. The final answer of an ENN is defined by the set of neurons that were activated.

Fig. 1 shows the overview of an ENN(3,3) in which 3 messages have been stored ([0,0,0], [0,2,2] and [2,2,0]) during the storage step. Let consider this network must decode the message [*?*,2,0] where the first symbol is unknown (i.e. the neuron in the cluster $C_0$ is unknown). The neurons $n_{(2,0)}$ (square-2) and $n_{(0,0)}$ (square-0) will remain active at the end of the decoding step since they are both connected to the neurons $n_{(2,1)}$ (round-2 in $C_1$) and $n_{(0,2)}$ (triangle-0 in $C_2$). Therefore, in this example, the false clique/message is [0,2,0] while the searched clique/message is [2,2,0]. The four notable properties of a binary associative memory are:

- The *global density*, i.e. the percentage of weights that are set to 1 in the network.
- The *degree,* i.e. the number of the weights that are set to 1 for a given neuron.
- The *memory size* $|\mathcal{W}|$, i.e. the total number of synaptic weights in the network which is equal to the size of the memory in bits to store those weights:

$$|\mathcal{W}| = \frac{L^2 * C * (C-1)}{2} \quad (3)$$

- The *capacity* $Cap$, i.e. the maximal number of messages a network can store and retrieve[1]:

$$Cap = \frac{(C-1) * N^2}{2 * C^2 \log_2\left(\frac{N}{C}\right)} \quad (4)$$

More details can be found in [8] for the model and in [25] for the fully-binary model. The ENN has a storage capacity of the order $N^2/(log\ N)^2$ messages for $N$ neurons, while the standard Hopfield model has a capacity of $N/(2\ log\ N)$ when the messages are *independent and identically distributed* (*i.i.d*).

### B. *Performance evaluation*

Since our goal is to proposed more performant ENN-inspired model, we need to define a metric for fair comparisons between models. Hence, to measure the *performance* of an ENN a *Retrieval Error Rate* (RER, i.e. the number of successful retrievals over the total number of trials) is used. In this paper, each point in RER results, is the average of 5000 storing/decoding error rates for a given number of data (with average deviation when it is relevant).

In an associative memory, the failure of a retrieval happens because some stored messages share the same set of values for the same symbols. In an ENN, this comes when several neurons from one cluster are connected to the same neuron in other clusters. Hence, when the decoding step starts, if that cluster is erased (i.e. it contains no active neurons), then several neurons might remain active in that cluster when the decoding step stops. This means that either all the remaining active neurons are part of the stored cliques (which look like the searched clique i.e. *true ambiguities*) or they are part of a blend between stored and not stored cliques (i.e. *false cliques* [8]).

The RER of the ENN depends on the number of stored messages and the distribution of those messages [20]. On one side, the RER increases with the number of stored messages. On the other side, for the same number of stored messages, the ENN gets a lower RER when those messages are uniformly distributed, compared to a non-uniform distribution (e.g. Gaussian distribution).

Fig. 2 shows the RER of an ENN (8,32) where ENN-Uniform is a network that memorized messages with a uniform distribution, while ENN-Gaussian is a network that memorized messages with a Gaussian distribution (mean 16, standard deviation 5). For 500 stored messages, and an erasure set to 50%, the RER of ENN-Uniform is 30% while the RER of ENN-Gaussian is 94%. As mentioned before, each point of these curves is the mean value of 5000 MATLAB simulations. In the

[1] The *capacity* is called *diversity* in [9].

sections V and VI, all the RER curves have been generated by using the same method.

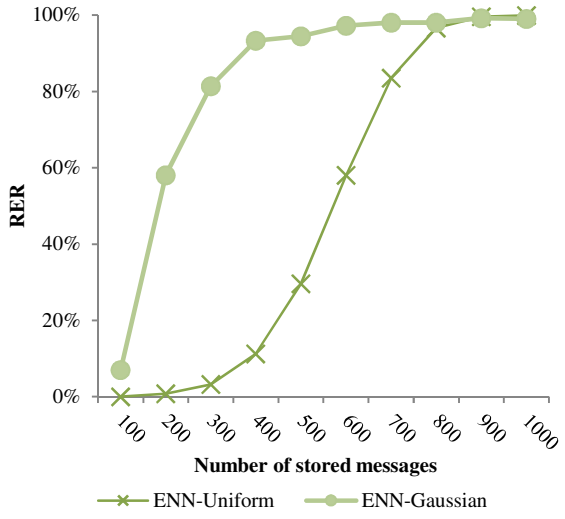



Fig. 2 RER evaluation of an ENN (8, 32) for different message distributions

### *C. State-of-the-art approaches to handle non-uniform distributions in ENNs*

Two solutions are proposed in [20] to improve performance of ENNs for non-uniform distributions of stored messages. These methods consider messages that are binary vectors and each symbol is a binary sub-vector. Before their storage, messages are recoded following different processes.

The first approach, named *adding bits,* consists in adding a group of bits to each symbol of the message to store. These added bits can either be: (1) randomly generated or (2) the least recently used sequence. With this solution the symbols to be stored become larger. As a consequence, the corresponding ENN is bigger to store the same amount of messages (de facto, more neurons and clusters are required to store the messages).

The second approach, named *Huffman,* consists in performing Huffman encoding [26] of the messages. Huffman encoding is a lossless data compression technique that consists in defining a codeword for each value that the symbols can take depending on the distribution. This method requires to determine the frequency (or the probability) of occurrence of the values from the set of messages to memorize. The sizes of the codewords depend on the values: the most frequent values have reduced sizes compared to the less frequent ones. The codewords are stored in an additional dictionary. Hence during the storing step the messages are encoded, and ENN stores these codewords instead of the messages. As the messages (i.e., the codewords) can have different sizes, the latter are extended to the size of the longest codeword by adding random bits.

A first improvement of the ENN models for non-uniform distributions without requiring any offline storage (i.e., with no dictionary), has been proposed by using clone neurons [10]. In this work, the clones were assigned before any storage (i.e., static allocation was considered). Hence, multiple instances of the same neuron exist in the network and are associated with the value of the symbol related to that neuron. Therefore, the degree a *classical* neuron would have is now distributed amongst its clone instances. The clones have a lower degree compared to neurons they are associated with. The proposed model competes with *Huffman* for the same memory cost (the extra-cost of the *Huffman* dictionary being excluded). This concept of clones has been reused in [22] through neuron duplication to improve the work from [20].

In the next section, Clone-based Neural Networks are introduced, formally defined and detailed.

## IV. Clone-based Neural Networks: Definitions And Principles

### *A. Basic definitions*

**Definition 1:** A *clone* is an instance of a neuron. A neuron may have one or several clones.

**Definition 2:** A *bundle* is a subset of clones of the same neuron. A neuron may be associated with one or several bundles.

**Definition 3:** A *sub-network* regroups a bundle of each neuron in each cluster. In other words, each cluster contains sets of clones instead of neurons. Each neuron is associated with one bundle in each sub-network exactly. the CbNN may contain one or several sub-networks.

The CbNN is defined by a set of parameters (see Table 1):

- The number of sub-networks $K$,
- The number of clusters for each sub-network $C$,
- The number of neurons associated with each cluster $L$,
- The number of clones representing those neurons in each cluster $\gamma$.

In order to synthetize the notation, all these parameters are combined in a single relation $\mathcal{R}(K, C, L, \gamma)$ allowing to define all the CbNN variants.

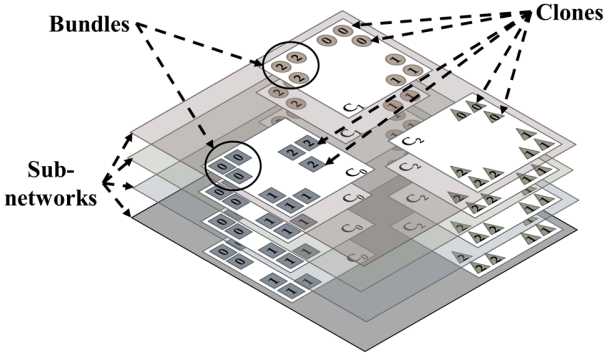



Fig. 3 An example of a CbNN $\mathcal{R}(4,3,3,9)$

Fig. 3 shows an overview of a CbNN $\mathcal{R}(4,3,3,9)$ containing 4 sub-networks. Each sub-network has 3 clusters ($C_0$, $C_1$, $C_2$). Every cluster contains 3 neurons and each neuron may be instantiated by any clone of its bundle. In this example, each bundle holds 3 clones. Therefore, each cluster contains $C*L$=3*3=9 clones and each neuron may be represented by $K*(\gamma/L)$=4*(9/3)=4*3=12 clones.

A CbNN $\mathcal{R}(K, C, L, \gamma)$ has $K$ sub-networks, each one containing $C$ clusters of $L$ neurons (or bundles of clones, if $\gamma >$

$L$). The total number of clones is thus $K * \gamma * C$. Each clone can be connected to $\gamma * (C-1)$ distant clones located in the same sub-network it belongs to. Therefore, by taking into account the symmetry of those connections, the total number of weights $\mathcal{W}$ in this network (equal to the cost to store the adjacency matrix in bits) is:

$$|\mathcal{W}| = \frac{K * \gamma^2 * C * (C-1)}{2} \quad (5)$$

TABLE 1 Notations used in the context of the CbNNs

| Symbol | Quantity |
|---|---|
| $K$ | Number of sub-networks |
| $C$ | Number of clusters |
| $L$ | Number of neurons per cluster |
| $\gamma$ | Number of clones per cluster |
| $\mathcal{R}(K,C,L,\gamma)$ | CbNN with $K$ sub-networks |
| $S_k$ | Subnetwork $k$ of $\mathcal{R}$ with $0 \leq k < K$ |
| $\mathcal{C}_{(j,k)}$ | Cluster $j$ of the sub-network $k$ |
| $n^x_{(i,j,k)}$ | Clone $x$ of the neuron $n_{(i,j)}$with $n^x_{(i,j,k)} \in \mathcal{C}_{(j,k)}$ |
| $w\left(n^x_{(i,j,k)};\ n^{x'}_{(i',j',k)}\right)$ | Synaptic weight between two clones, $n^x_{(i,j,k)}$ and $n^{x'}_{(i',j',k)}$ |
| $N_{(j,k)}$ | Set of clones in the cluster $\mathcal{C}_{(j,k)}$ |
| $\Gamma\left(n_{(i,j)}\right)$ | Set of bundles of the neuron $n_{(i,j)}$ |

### B. Properties

**Property 1.** If a network possesses $K$ sub-networks then,

$$\forall i,j, \max\left(\left|\Gamma\left(n_{(i,j)}\right)\right|\right) = K, with\ K \geq 1$$

In other words, for a neuron, its number of bundles is at least equal to 1 and at most equal to the number of sub-networks.

**Property 2.** If $b \subset \Gamma\left(n_{(i,j)}\right)$ then,

$$|b| \geq 0$$

In other words, a bundle $b$ of a neuron may be empty and thus contains no clone.

**Property 3.** $w\left(n^x_{(i,j,k)};\ n^{x'}_{(i',j',k')}\right) = \begin{cases} 1\ or\ 0\ si\ k = k' et\ j \neq j' \\ 0, else \end{cases}$

In other words, two clones can only be connected if they belong to the same sub-network ($k = k'$) while being in different clusters ($j \neq j'$).

**Property 4.** A sub-network is an entity able to store and decode messages.

**Property 5.** If a network possesses $K$ sub-networks, then a neuron $n_{(i,j)}$ is associated with $K$ possible values: one value $v(n_{(i,j)};k)$ for each sub-network $S_k$ where $v\left(n_{(i,j)};k\right)$ is the state value of the neuron $n_{(i,j)}$ according to the sub-network $S_k$.

### C. Taxonomy of the CbNNs

We define two main classes of the CbNNs which differ according to the way clones and sub-networks are allocated: *Static* or *Dynamic*. In a $\mathcal{R}(K,C,L,\gamma)$ K and/or $\gamma$ may be statically defined when the network is designed, or in dynamic model, the network is designed in order to be able to adapt these parameters, depending on the sets of messages to store. The different combinations of these degree of freedom are gathered in Table 2.

TABLE 2 Variants of the CbNNs and their classification

| | | Number of sub-networks ($K$) | | |
|---|---|---|---|---|
| | | **1** | **Static allocation** | **Dynamic allocation** |
| Number of clones per bundle ($\gamma$) | **1** | $\mathcal{R}(1,C,L,L)$ ⇔ ENN (see [8]) | $\mathcal{R}(K_S,C,L,L)$ (see [10]) | $\mathcal{R}(K_D,C,L,L)$ |
| | **Static allocation** | $\mathcal{R}_U(1,C,L,\gamma_S)$ ⇔ *adding bits* (see [8])<br>$\mathcal{R}_{NU}(1,C,L,\gamma_S)$ ⇔ *Huffman* (see [20]) | $\mathcal{R}_U(K_S,C,L,\gamma_S)$<br>$\mathcal{R}_{NU}(K_S,C,L,\gamma_S)$ | $\mathcal{R}(K_D,C,L,\gamma_S)$ |
| | **Dynamic allocation** | $\mathcal{R}(1,C,L,\gamma_D)$ ($if\ \max(\gamma_D)=2$ ⇔ *twin-neurons* [22]) | $\mathcal{R}(K_S,C,L,\gamma_D)$ | $\mathcal{R}(K_D,C,L,\gamma_D)$ |

As an example, $\mathcal{R}(1,C,L,L)$, which is a static model since $K_S$=1 and $\gamma_S$=L, corresponds to the original ENN model from [8]. As another example, $\mathcal{R}(K_D,C,L,\gamma_D)$ indicates that the number of sub-network and the number of clones in the bundles may be *dynamically* increased (i.e. the number of sub-networks and the number of clones in bundles may change on-the-fly, depending on dedicated storage algorithms).

Table 2 reviews the different variants of the CbNNs. The networks with the index "*U*" (resp. "*NU*") represent models where bundles contain a fixed number of clones *Uniformly* (resp. *Non-Uniformly*) allocated to each neuron. These allocation strategies will be further detailed in subsection IV.B.

### D. Storing step and retrieving procedure

#### 1) Storing messages in a CbNN

This section introduces the main principles of the storage procedure. More details are provided in sections IV and V for each CbNN variants. Storing messages in a CbNN is performed in four steps:

1. The neurons (i.e., all the corresponding bundles of clones) related to the symbols of the message, are activated.
2. A unique sub-network is selected (since a neuron may have several bundles distributed over several sub-networks).
3. A unique active clone is selected in each cluster (since a cluster in a sub-network may contain several clones for the same neuron).
4. The weights between the selected clones are set to '1'.

The selection of a sub-network may be done randomly or by competition. The competition between clones (resp. between sub-networks) is realized by computing a score for the activated clones (resp. sub-networks) and by choosing the clones (resp. sub-network) having the best score i.e. the lowest one. Different approaches have been explored to compute this score and the most efficient ones are presented later in this paper (see the sections V and VI).

*2) Decoding procedure to retrieve a message*

The decoding step allows to retrieve a stored message when some of the input message symbols are erased.

This procedure is iterative and performs in four steps:

1. Given the input partially erased message: for each message symbol if the value is known, the associated clone is activated, otherwise all the clones are activated. Thus, in each cluster either one or all the clones are activated.
2. The score of each clone in the network is computed by using the equations (1) and (2) (see section II).
3. The state of the selected neurons, in each sub-network is determined. A neuron is active in a sub-network, if at least one of its active clones is contained in that sub-network.
4. Each sub-network is evaluated to determine if it *converged*. A sub-network converged once the previous steps have been applied when each of its clusters possesses exactly one active neuron i.e. each cluster contains a unique bundle having itself one active clone.

A CbNN performs a *successful decoding* if at the end of the decoding process, there is at least one sub-network that has converged. If several sub-networks have converged, one of them is randomly selected.

## V. Static CbNNs

In static variants of the CbNNs, the number of clones allocated to each neuron and the number of sub-networks in the CbNN are fixed (i.e., constant). The storage algorithm thus selects the "best" clones or sub-networks from a completely predefined set. For every static variant of the CbNNs, we propose two algorithms: the arbitrary or Random (RND) algorithm and the Least Dense Selection (LDS) algorithm.

The RND algorithm consists in:

1. *Randomly* choosing a sub-network,
2. Activating the clones associated with the symbols of the input message,
3. *Randomly* selecting one active clone per cluster in that sub-network,
4. Setting to 1 the weights between the selected clones.

The LDS algorithm consists in:

1. Computing a score (see (6)) for each sub-network,
2. *Randomly selecting* one of the sub-networks with the best score, i.e. the lowest one (see (7)).

The first step of the LDS involves two sub-steps. First, one active clone per cluster is randomly activated, while the others are deactivated. In fact, this random selection is still used in this sub-step to avoid a huge amount of local score computations[2]. An arbitrary selection still offers good results as shown in the experiments. The second sub-step consists in computing the score of each sub-network which is equal to the sum of the scores of its active clones.

Let $n^{x}_{(*,j,k)}$ be an active clone, its score $sc\left(n^{x}_{(*,j,k)}\right)$ is:

$$sc\left(n^{x}_{(*,j,k)}\right) = \sum_{j'=0,j'\neq j}^{C-1}\left(\sum_{\boldsymbol{n}^{x'}_{(i',j',k)}\in \boldsymbol{N}(\boldsymbol{j'},\boldsymbol{k})}\left(v\left(n^{x'}_{(i',j',k)}\right)\oplus w\left(n^{x}_{(*,j,k)};\, n^{x'}_{(i',j',k)}\right)\right)\right) \quad (6)$$

The score $sc(k)$ of a sub-network $k$ is given by:

$$sc(k) = \sum_{j=0}^{C-1}\left(sc\left(n^{x}_{(*,j,k)}\right)\right) \quad (7)$$

The selected sub-network $k_l$ which stores the message is one of the sub-networks having the lowest score $sc(k_l)$:

$$sc(k_l) = min\left(\left(sc(k)\right)_{0\leq k<K}\right) \quad (8)$$

This competition tends to choose the sub-network that will have the lowest degrees for its selected active clones once the message has been stored. It globally tries to keep as low as possible clones' degrees.

### A. Original ENN model $\mathcal{R}(1, C, L, L)$

$\mathcal{R}(1, C, L, L)$ contains a unique sub-network $\left(\forall i, j, \left|\Gamma\left(n_{(i,j)}\right)\right| = 1\right)$ in which a unique clone is allocated in each bundle $\left(\forall i, j, \forall p \in \Gamma\left(n_{(i,j)}\right), |p| = 1\right)$. In other words, we have a unique sub-network in which each neuron only possesses a unique clone.

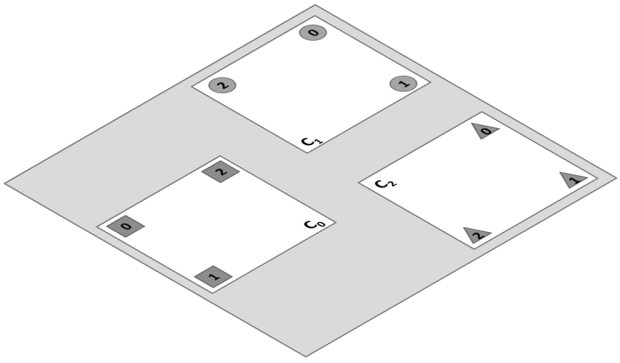


Fig. 4 An example of a $\mathcal{R}(1,3,3,3)$

Fig. 4 shows a $\mathcal{R}(1,3,3,3)$ having a unique sub-network containing 3 clusters, each cluster with 3 clones and a single clone being allocated to each neuron.

### B. CbNN variant $\mathcal{R}(1, C, L, \gamma_S)$

$\mathcal{R}(1, \mathrm{C}, \mathrm{L}, \gamma_S)$ contains a unique sub-network $\left(\forall i, j, \left|\Gamma\left(n_{(i,j)}\right)\right| = 1\right)$ in which a fixed number of clones is allocated to each bundle $\left(\forall i, j, \forall p \in \Gamma\left(n_{(i,j)}\right), |p| = q\right)$.

Fig. 5 shows an example of a $\mathcal{R}(1,3,3,9)$ having a unique sub-network containing 3 clusters, each cluster with 3 neurons represented by a total of 9 clones per cluster.

[2] Indeed, if clusters contain several active clones, in order to perform an efficient local competition, the score of each active clone has to be computed for every combination of storage (i.e. for all possible weight activation generated by the message to store). Hence, if $\frac{\gamma}{L}$ clones are allocated to each neuron in the network, then the total number of combinations is $\left(\frac{\gamma}{L}\right)^{C}$…

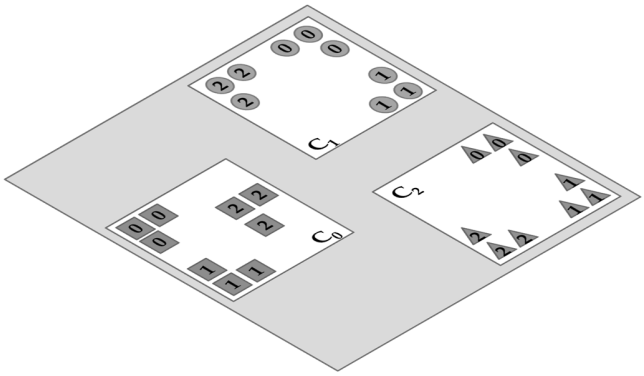


Fig. 5 An example of a $\mathcal{R}$ (1,3,3,9)

Here, since there is only one sub-network, $K = 1$. Its memory size, given by eq. (3), is:

$$|\mathcal{W}| = \frac{\gamma^2 * (\mathrm{C} - 1) * C}{2} \quad (9)$$

The number of clones allocated to each neuron may be the same for every neuron (Uniform allocation, $\mathcal{R}_U$) or may depend on the distribution (Non-Uniform allocation, $\mathcal{R}_{NU}$).

*1) Uniform allocation $\mathcal{R}_U$*

In the case of the *Uniform* allocation, the same number of clones is allocated to each neuron. Then, with such constraint the model is similar to the *adding bits* approach introduced in [20].

*2) Non-Uniform allocation $\mathcal{R}_{NU}$*

In the case of the *Non-Uniform* allocation, the number of clones allocated to a given neuron depends on the occurrence frequency of the symbol value associated with that neuron. Such frequency can be obtained by evaluating the distribution over a sample of messages coming from the set of messages to store.

Since this variant relies on occurrence frequency, it is similar to the *Huffman* approach proposed in [20]. However, instead of generating a dictionary of codes as in [20], we propose a simpler alternative by assigning a precise number of clones to each neuron based on two properties: the occurrence frequency of its related symbol value and the total number of available clones in its cluster.

Let $n_{(i,j)}$ be a neuron and $F_{(i,j)}$, the occurrence frequency of its associated value. So, the number of clones $\beta(n_{(i,j)})$ allocated to $n_{(i,j)}$ is:

$$\beta(n_{(i,j)}) = \left\lfloor F_{(i,j)} * L * \left(\frac{\gamma}{L} - 1\right) \right\rfloor + 1 \quad (10)$$

Equation (10) ensures that each neuron has at least one clone. If some clones in a cluster remain unallocated, they are equally distributed over the most involved neurons in that cluster. Such fine grain allocation allows determining an optimized number of clones for each neuron which reduces the variation between the clone degrees in a given cluster.

### C. CbNN variant $\mathcal{R}(K_S, C, L, L)$

$\mathcal{R}(\mathrm{K_S}, \mathrm{C}, \mathrm{L}, \mathrm{L})$ contains a fixed number of sub-networks $(\forall i, j, |\Gamma(n_{(i,j)})| = K, K \geq 1)$ in which a unique clone is allocated in each bundle for every neuron $(\forall i, j, \forall p \in \Gamma(n_{(i,j)}), |p| = 1)$. This model is the one we have presented in [10].

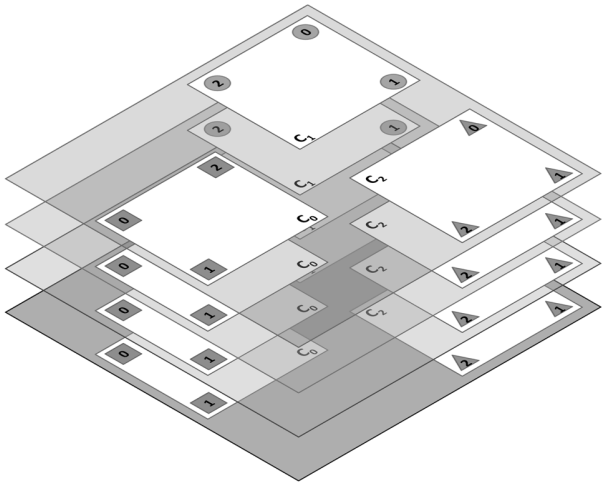


Fig. 6 An example of a $\mathcal{R}(4,3,3,3)$

Fig. 6 shows the example of a $\mathcal{R}(4,3,3,3)$ with 4 sub-networks, each one containing 3 clusters, each cluster with 3 neurons (one unique clone for each neuron). $\mathcal{R}(\mathrm{K_S}, \mathrm{C}, \mathrm{L}, L)$ gathers several ENNs within the same network. Its memory size, given by eq. (3), is:

$$|\mathcal{W}| = \frac{K * L^2 * (C - 1) * C}{2} \quad (11)$$

In this CbNN variant, the multiplication of the number of sub-networks allows to multiply by $K$ the capacity of the original network (ENN).

### D. CbNN variant $\mathcal{R}(K_S, C, L, \gamma_S)$

$\mathcal{R}(\mathrm{K_S}, \mathrm{C}, \mathrm{L}, \gamma_\mathrm{S})$ contains a fixed number of sub-networks $(\forall i, j, |\Gamma(n_{(i,j)})| = K, K \geq 1)$ in which a fixed number of clones is allocated in the bundles $(\forall i, j, \forall p \in \Gamma(n_{(i,j)}), |p| = q)$.

Fig. 3 shows an example of a $\mathcal{R}(4,3,3,9)$ having 4 sub-networks with 3 clusters each. Each cluster has 3 neurons represented by a total of 9 clones per cluster.

The total number of weights of $\mathcal{R}(K_S, C, L, \gamma_S)$ is given by (5). It is the generalization of the previously presented variants. Therefore, it possesses the same property in terms of allocation (Uniform allocation $\mathcal{R}_U(K_S, C, L, \gamma_S)$, or Non-Uniform allocation $\mathcal{R}_{NU}(K_S, C, L, \gamma_S)$). When $\mathcal{R}_U$ allocation is considered, an identical number of clones is allocated to each bundle while in $\mathcal{R}_{NU}$, the number of allocated clones depends on the message distribution. Using several sub-networks increases the number of messages that the network can store. The non-uniform allocation allows to adapt the CbNN to the message distribution.

### E. Capacity

In this section, the models previously introduced are compared in terms of capacity (see equation 4 in section II.A) i.e., the maximal amount of data that can be stored and then retrieved.

For $\mathcal{R}(1, C, L, L)$, the total number of neurons in the network is $N = L * C$, the capacity of this variant is given for i.i.d. messages by ([9]):

$$Cap(\mathcal{R}(1,C,L,L)) = \frac{(C-1)*N^2}{2*C^2*log_2\left(\frac{N}{C}\right)} = \frac{(C-1)*(L)^2}{2*log_2(L)} \quad (12)$$

For $\mathcal{R}(1,C,L,\gamma_S)$, we replace $N$ by the total number of clones in the sub-network which is $\gamma_S * C$. Then, its capacity is:

$$Cap(\mathcal{R}(1,C,L,\gamma_S)) = \frac{(C-1)*(\gamma_S)^2}{2*log_2(\gamma_S)} \quad (13)$$

For $\mathcal{R}(K_S,C,L,L)$, the number of sub-networks multiplies the capacity of $\mathcal{R}(1,C,L,L)$ by $K_S$. Then, it is equal to:

$$Cap(\mathcal{R}(K_S,C,L,L)) = K_S * \frac{(C-1)*(L)^2}{2*log_2(L)} \quad (14)$$

For variant $\mathcal{R}(K_S,C,L,\gamma_S)$, the number of sub-networks multiplies the capacity by $K_S$, while the number of clones in each sub-network is multiplied by $\frac{\gamma_S}{L}$ compared to $\mathcal{R}(1,C,L,L)$.

Then, its capacity is given by:

$$Cap(\mathcal{R}(K'_S,C,L,\gamma'_S)) = K'_S * \frac{(C-1)*(\gamma'_S)^2}{2*\log_2(\gamma'_S)} \quad (15)$$

In order to have the same memory cost (for $\mathcal{R}(1,C,L,\gamma_S)$, $\mathcal{R}(K_S,C,L,L)$ and $\mathcal{R}(K_S,C,L,\gamma_S)$), the following expression has to be verified:

$$K_S = \left(\frac{\gamma_S}{L}\right)^2 = K'_S * \left(\frac{\gamma'_S}{L}\right)^2 \quad (16)$$

Equation (16) is obtained by matching (5), (9) and (11). The comparison between those capacities (see (17) and (18)) shows that the capacity of $\mathcal{R}(K_S,C,L,1)$ is greater than the one of $\mathcal{R}(1,C,L,\gamma_S)$ since $\gamma_S > L$. It is also greater than the capacity of $\mathcal{R}(K'_S,C,L,\gamma'_S)$ but with a lower gain (since $\gamma_S > \gamma'_S$).

$$\frac{Cap(\mathcal{R}(K_S,C,L,L))}{Cap(\mathcal{R}(1,C,L,\gamma_S))} = \frac{K_S*L^2*\log_2(\gamma_S)}{{\gamma_S}^2*\log_2(L)} = \frac{\log_2(\gamma_S)}{\log_2(L)} \quad (17)$$

$$\frac{Cap(\mathcal{R}(K_S,C,L,L))}{Cap(\mathcal{R}(K'_S,C,L,\gamma'_S))} = \frac{K_S*L^2*\log_2(\gamma'_S)}{K'_S*{\gamma'_S}^2*\log_2(L)} = \frac{\log_2(\gamma'_S)}{\log_2(L)} \quad (18)$$

Therefore, $\mathcal{R}(K_S,C,L,L)$ models are able to store the greatest number of messages. Unfortunately, the capacity does not guarantee good results because of the distribution of the data to store.

### *F. Performances of the CbNNs with Static Clone Allocation*

This section presents efficiency comparisons of the different CbNN static variants in terms of RER.

#### *1) Experimental setup*

For fair comparisons with state of the art approaches, the number of clusters is set to 8 and the number of neurons is set to 32 for each cluster. The results are generated from simulations based on the decoding of messages with a non-uniform distribution. The distribution of the messages is Gaussian with a mean of 16 and a standard deviation of 5. Like in section II, each point of the curves is the mean value of 5000 MATLAB simulations of each variant. The experiments were performed with MATLAB 2014 from Mathworks. Every network has been configured to have the same memory cost equal to 16 ENNs(8,32) = 16 $\mathcal{R}$(1,8,32,32). The ratio 16 is the one used in [20] to compare different ENN approaches.

The RER evaluation is based on the decoding of messages, randomly selected. RER mean variation $\Delta$ over 5000 simulations is provided for all the results. The results recorded during our experiments are obtained with the same constraint used in [8] or [20]: 50% of the input messages to be retrieved are randomly erased. For each message to decode, 4 symbols over 8 are unknown and have to be retrieved. Four iterations were performed for each decoding, since this number is sufficient to achieve the best RER of the original ENN model ($\mathcal{R}(1,C,L,L)$ in this paper) [8].

In the figures, the best error ratio of the ENN (with Gaussian distribution of the messages) are given as references. Table 3 shows the parameters in terms of dimensions for the evaluated variants.

TABLE 3 Dimensions of the CbNNs with static clone allocation

| Variants | $K_S$ | $\gamma_S$ | $\frac{\gamma_S}{L}$ |
|---|---|---|---|
| $\mathcal{R}(1,C,L,\gamma_S)$ | 1 | 128 | 128/32=4 |
| $\mathcal{R}(K_S,C,L,L)$ | 16 | 32 | 32/32=1 |
| $\mathcal{R}(K_S,C,L,\gamma_S)$ | 4 | 64 | 64/32=2 |

Several ENNs with static clone allocation were exposed in the previous sub-sections V.A to V.E. Each one has different configurations depending on the storage algorithms ($\mathcal{R}(K_S,C,L,L)$ and $\mathcal{R}(K_S,C,L,\gamma_S)$) or the allocation strategy ($\mathcal{R}(1,C,L,\gamma_S)$ and $\mathcal{R}(K_S,C,L,\gamma_S)$). In order to avoid overloaded figures, comparisons are progressively performed by assessing different configurations for the same variant, and we finally compare the best static variants.

#### *2) Performance of $\mathcal{R}(1,C,L,\gamma_S)$*

Fig. 7 shows the RER evolution of this variant for different uniform and non-uniform allocation approaches. $\mathcal{R}_{NU}(1,C,L,\gamma_S)$ obtains the best results thanks to its clone allocation policy allowing adaptation to data distribution. Moreover, we observe a RER < 1,2% up to 1500 messages ($\Delta \pm 0.4\%$).

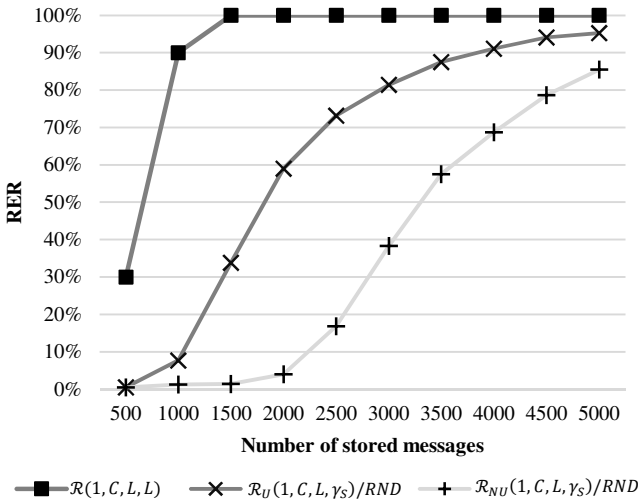



Fig. 7 RER of $\mathcal{R}(1,C,L,\gamma_S)$ depending on the allocation strategy of the clones

#### *3) Performance of $\mathcal{R}(K_S,C,L,L)$*

Fig. 8 shows the efficiency of $\mathcal{R}(K_S,C,L,L)$ for the storage algorithms RND and LDS. $\mathcal{R}(K_S,C,L,L)$/LDS gets the best results with an RER < 2.1% ($\Delta \pm 1.1\%$) up to 1500 messages. This is due to the LDS algorithm that tends to slow down the

increase of the clones' degrees during the storage step, compared to an arbitrary clone selection.

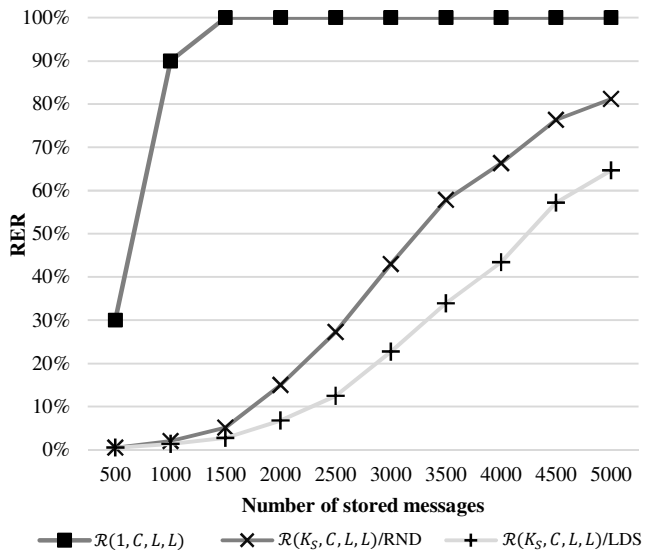



Fig. 8 The RER of $\mathcal{R}(K_S,C,L,L)$ depending on the memorization algorithms

*4) Performances of $\mathcal{R}(K_S,C,L,\gamma_S)$*

Fig. 9 shows the RER evolution of $\mathcal{R}(K_S,C,L,\gamma_S)$ for different allocation approaches and storage algorithms. $\mathcal{R}_{NU}(K_S,C,L,\gamma_S)$/LDS achieves a significant performance improvement with an RER < 1,2% ($\Delta \pm 0.3\%$) up to 1500 messages thanks to the adaptive allocation policy and the use of the LDS method.

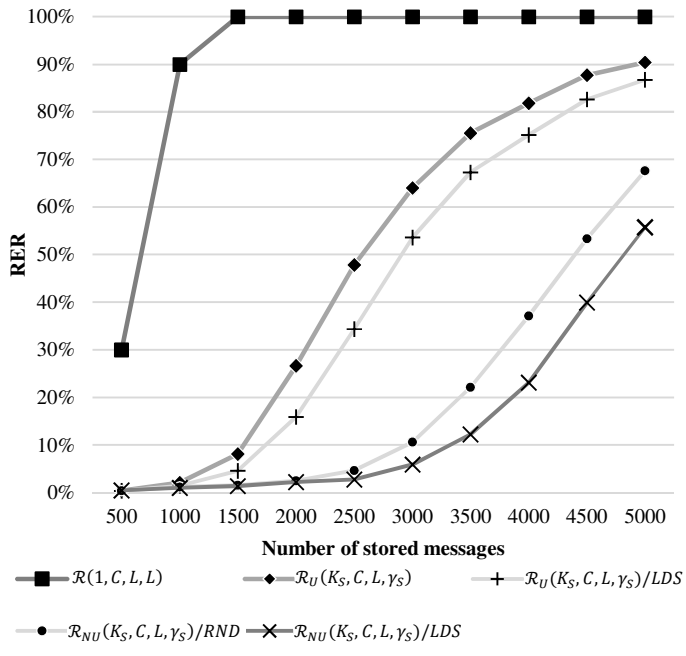



Fig. 9 The RER of $\mathcal{R}(K_S,C,L,\gamma_S)$ depending on the clone allocation strategies and the storage algorithms (RND vs. LDS)

*5) Comparison of the best static CbNN variants*

Fig. 10 compares the accuracy of the best CbNN variants. $\mathcal{R}_{NU}(K_S,C,L,\gamma_S)$/LDS still gets the best results (RER < 1,2% ($\Delta \pm$ 0.3%) up to 1500 messages) thanks to its smart clone allocation (based on the use of several sub-networks) and a smart storage algorithm.

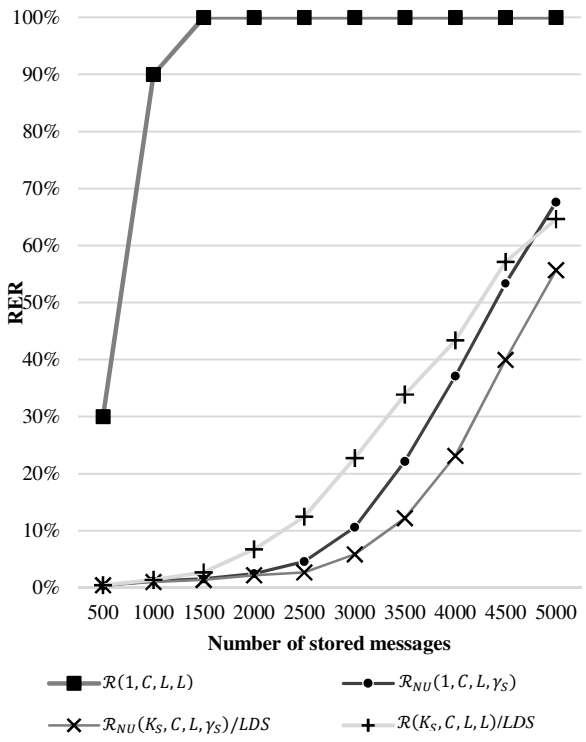



Fig. 10 The RER of the best networks with static clone allocation

## VI. DYNAMIC CBNNS

As previously introduced, dynamic allocation is based on the idea that a CbNN is designed to optimize the number of sub-network $K$ and/or the number of clones in a cluster $\gamma_i$. In other words, clones and/or sub-networks can be allocated during the storing step depending on the message distribution. In practice, to remain realistic, the total amount of memory available in the network is bounded (i.e., infinite clone allocation is forbidden). Hence, the maximal number of clones per cluster but also the maximal number of sub-networks are limited and can vary from 0 up to the maximum (i.e., resp. $K_D$ or $\gamma_D$).

As a result, some dynamic variants behave rather similarly to some static variants and are not presented in this paper due to space limitation. Hence, in $\mathcal{R}(K_D,C,L,\gamma_S)$, the number of sub-networks may grow while each bundle contains a fixed number of clones. If the number of sub-networks is limited, then this variant is roughly similar to a $\mathcal{R}(K_S,C,L,\gamma_S)$ since the latter contains a constant number of sub-networks.

In $\mathcal{R}(K_S,C,L,\gamma_D)$, the number of sub-networks is fixed and the number of clones per bundle can increase. Since the number of clones is limited, this variant becomes rather similar to $\mathcal{R}(K_S,C,L,\gamma_S)$ which contains a constant number of sub-networks.

Finally, in $\mathcal{R}(K_D,C,L,L)$, the number of sub-networks may be increased while every bundle contains a unique clone. Since the number of sub-networks is limited, this variant is roughly

similar to $\mathcal{R}(K_S, C, L, L)$ which contains a fixed number of sub-networks.

Therefore, only two variants are thoroughly presented in this section: $\mathcal{R}(1, C, L, \gamma_D)$ and $\mathcal{R}(K_D, C, L, \gamma_D)$. As a first step, a storage algorithm compatible with both models is introduced. The models are next presented.

### A. Storage algorithm for dynamic CbNNs

Two rules are required for this storage algorithm.

***Rule 1*** *The allocation of a clone to an active neuron may be performed if and only if there is an available clone in its cluster.*

If this constraint is verified, the allocation of a clone to an active neuron is effectively performed when: either (1) there is no clone allocated to the neuron, or (2) the degree of the Most Recently Used clone of the neuron is higher than a threshold $\alpha$ called *degree limit*. Hence, by definition, our dynamic models are based on non-uniform allocation of clones.

***Rule 2*** *The allocation of a new sub-network to the network is performed:*

*(1) If the allocation of a clone to an active neuron could not be performed in the Last Used Subnetwork*

*and*

*(2) If the maximal number of usable sub-networks is not reached.*

The storage process is defined as follows: as soon as the network receives a message to store, the neurons and the clones are activated in the most recently used sub-network.

For each cluster in the sub-network, an active clone has to be selected. For a given cluster $C_x$, there are two cases:

1. *There is no active clone in* $C_x$ (no clone was allocated to the active neuron in $C_x$). The allocation is performed, and the new clone is used to store the message.
2. *There is at least one active clone in* $C_x$. In that case, if the degree of the last used clone is higher than $\alpha$, a new clone is allocated to store the message. Otherwise, the most recently used clone is selected (Rule 1).

If a clone allocation has to be performed in a cluster and if there is no available clone, then a new sub-network is added, if possible (Rule 2). This new sub-network becomes the most recently used sub-network and the whole process is restarted.

If a sub-network cannot be added to the network, our approach performs either RND or LDS algorithms (depending on user requirements) without any modification (see Section V):

1. *With RND*, a sub-network $S_x$ is randomly chosen, relevant clones are activated in $S_x$ and an active clone is randomly chosen in each cluster.
2. *With LDS*, for each sub-network, an active clone is randomly selected in each cluster and the scores are computed (see section V). If a cluster in a sub-network does not contain any active clone, the sub-network is excluded from the competition since it cannot help to create a full neural clique.

The value of $\alpha$ can be derived from the notion of neuron degree for a given number of messages to store $M$. If we consider *i.i.d.* messages, the degree limit $\alpha$ is computed as follows:

$$\alpha = 1 - \left(1 - \frac{1}{\gamma}\right)^{\frac{M}{K}} \tag{19}$$

Equation (19) does not always allow to reach the best performance, in particular for a high number of stored messages. Indeed, this heuristic is based on uniform message distribution while *i.i.d* messages are considered.

### B. CbNN variant $\mathcal{R}(1, C, L, \gamma_D)$

$\mathcal{R}(1, C, L, \gamma_D)$ contains a unique sub-network $(\forall i, j, |\Gamma(n_{(i,j)})| = 1)$ in which the number of clones in bundles may dynamically increase $(\forall i, j, \forall p \in \Gamma(n_{(i,j)}), |p| \leq \gamma)$.

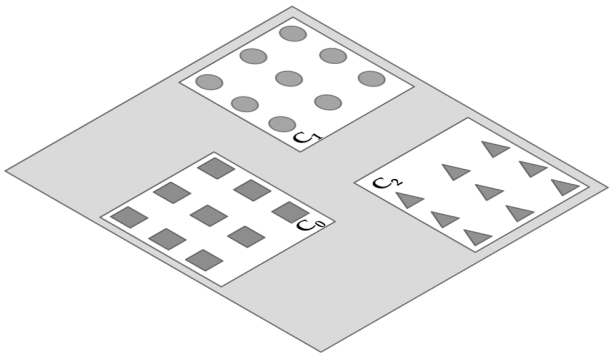


Fig. 11 An example of a $\mathcal{R}(1,3,3,9)$

Fig. 11 shows an example of a $\mathcal{R}(1,3,3,9)$ having a unique sub-network containing 3 clusters with each cluster with 9 clones not yet allocated to a dedicated neuron. On average, 3 clones may be allocated to each neuron.

### C. CbNN variant $\mathcal{R}(K_D, C, L, \gamma_D)$

In $\mathcal{R}(K_D, C, L, \gamma_D)$, both the number of sub-networks $(\forall i, j, |\Gamma(n_{(i,j)})| \leq K_D, with\ K_D \geq 1)$, and the number of clones in each bundle $(\forall i, j, \forall p \in \Gamma(n_{(i,j)}), |p| \leq \gamma_D)$ may be increased.

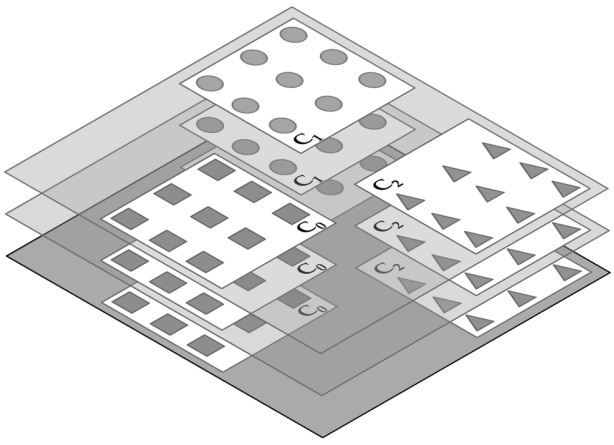


Fig. 12 An example of a $\mathcal{R}(4,3,3,9)$

Fig. 12 shows a $\mathcal{R}(4,3,3,9)$ having 3 sub-networks (over the 4 possible sub-networks), containing 3 clusters, with each cluster

having 9 clones (the maximum). On average, 3 clones may be allocated to each neuron in each sub-network.

### *D. Capacity of the CBNNs with dynamic clone allocation*

The capacity is only defined for a uniform distribution of messages. Therefore, since each clone allows to memorize the same quantity of information (for any allocation policy), the capacity of those networks is identical to their static counterpart ($\mathcal{R}(K_D, C, L, \gamma_D) \Leftrightarrow \mathcal{R}(K_S, C, L, \gamma_S)$, $\mathcal{R}(1, C, L, \gamma_D) \Leftrightarrow \mathcal{R}(1, C, L, \gamma_S)$).

### *E. Performances of the CbNNs with Dynamic Clone Allocation*

The following simulations have been performed with the same experimental setup presented in section V. Table 4 shows the configurations in terms of dimensions for the different variants of the CbNNs with dynamic clone allocation.

TABLE 4 Dimensions of the evaluated CbNN variants with dynamic clone allocations

| Variant | K | $\gamma_D$ | $\frac{\gamma_D}{L}$ |
|---|---|---|---|
| $\mathcal{R}(1, C, L, \gamma_D)$ | 1 | 128 | 128/32=4 |
| $\mathcal{R}(K_D, C, L, \gamma_D)$ | 4 | 64 | 64/32=2 |

As for the static variant experiments, each point of the curves is the mean value of 5000 MATLAB simulations. For the sake of clarity, comparisons are progressively performed for each variant depending on storage algorithms (RND and LDS) and degree limit $\alpha$. Finally, a comparison between the best networks with dynamic clone allocation is provided.

*Nota bene*: for these experiments, we have tested two $\alpha$ values: an arbitrary value $\alpha$ =5%, and $\alpha$ = *opt* which is the degree limit obtained by using (19).

#### *1) Performance of* $\mathcal{R}(1, C, L, \gamma_D)$

Fig. 13 shows the performance of $\mathcal{R}(1, C, L, \gamma_D)$ for different degree limits $\alpha$. It can be observed that $\mathcal{R}(1, C, L, \gamma_D)(\alpha$=5%) obtains the best performance: RER < 2% ($\Delta \pm 0.6\%$) up to 1000 messages. The value $\alpha = 5\%$ results in a "faster" allocation of the clones. This comes from the fact that some neurons have no allocated clones. Therefore, some messages that should have been stored, are not. For instance, over 5000 messages, 100 messages (2%) were not stored by the $\mathcal{R}(1, C, L, \gamma_D)(\alpha$=5%). Non-stored messages may impact performance positively or negatively, since the network becomes specialized in storing and retrieving messages with the most frequent neuron state values.

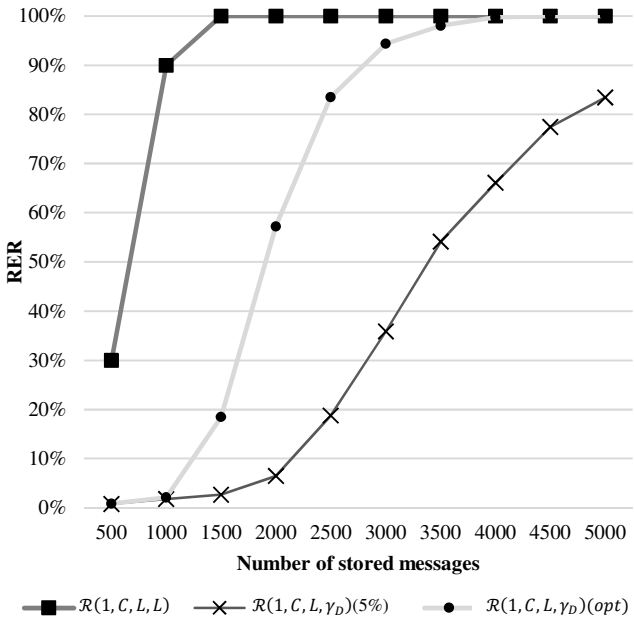



Fig. 13 RER of $\mathcal{R}(1, C, L, \gamma_D)$ depending on the density limit

#### *2) Performance of* $\mathcal{R}(K_D, C, L, \gamma_D)$

Fig. 14 shows the RER evolution of $\mathcal{R}(K_D, C, L, \gamma_D)$ for different storage algorithms and density limits. In overall, $\mathcal{R}(K_D, C, L, \gamma_D)(5\%)$/LDS offers the best results compared to the other $\mathcal{R}(K_D, C, L, \gamma_D)$ configurations presented in this case study.

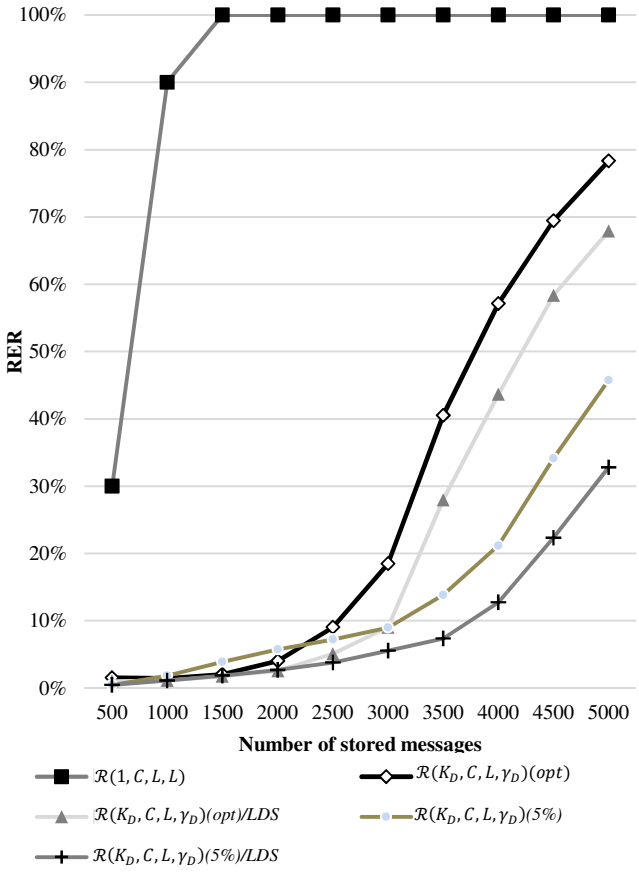



Fig. 14 The RER of $\mathcal{R}(K_D, C, L, \gamma_D)$ networks depending on the storage algorithms and the density limits

It can be observed that there is no real advantage to use $\mathcal{R}(K_D, C, L, \gamma_D)$(opt)/LDS compared to $\mathcal{R}(K_D, C, L, \gamma_D)$(5%)/LDS. The RER of the first one is lower than 1,5% up to 1500 messages (Δ±0.5%), while the second has a RER lower than 2.4% (Δ±0.7%). However, Fig. 14 shows that the results are not good above 2000 messages. This is due to the parameter $\alpha$ computed with equation (19) by considering constants values while messages are *i.i.d* in these experiments. Instead, by using a lower density limit (e.g. 5%), more clones are allocated since the allocation threshold is lower. Moreover, the LDS algorithm is preferably applied for a larger number of messages in order to achieve good performance.

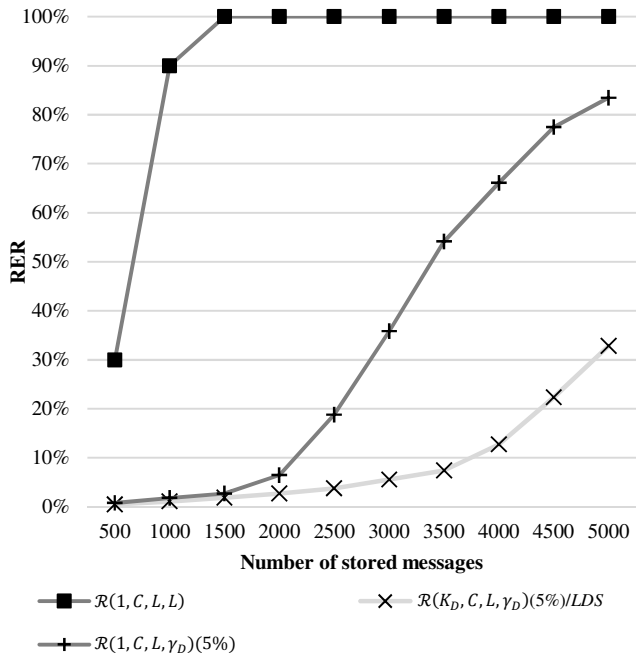



Fig. 15 The RER of the best networks with dynamic clone allocations

*3) Comparing the best CbNN variants with dynamic clone allocation*

Fig. 15 compares the accuracy of the best networks with dynamic clone allocation. $\mathcal{R}(K_D, C, L, \gamma_D)$(5%)/LDS clearly outperforms $\mathcal{R}(1, C, L, \gamma_D)$. This improvement (up to 55%) comes from the use of multiple sub-networks and the use of the smart storage algorithm LDS.

## VII. Comparing Static vs. Dynamic CbNN variants

In this section, the RER of the best static and dynamic variants are compared. First a worst case erasure rate of 50% (i.e., half of the input messages to be retrieved are randomly erased) is considered like in [8] or [20]. This strong constraint has been set in order to fairly evaluate the performances of our models compared to the original ENN. In order to explore the interest of our static and dynamic models, we also performed these experiments with a more classical message erasure of 25% (like in [12][11][10]…).

### A. Erasure rate of 50%

In Fig. 16, it can be observed that $\mathcal{R}(K_D, C, L, \gamma_D)$ (5%)/LDS and $\mathcal{R}_{NU}(K_S, C, L, \gamma_S)/LDS$ achieves almost the same efficiency for less than 2000 messages. The proposed dynamic CbNN seems to be more interesting when more messages need to be stored, RER < 1,8% (Δ±0,08%) up to 2500 messages.

### B. Erasure rate of 25%

In Fig. 17, the RER of the best static and dynamic CbNN variants is reported, with an erasure rate of 25%, and it is compared with their previous results from Fig. 17. The original ENN model $\mathcal{R}(1, C, L, L)$, which has the same trends than in Fig. 16, is not reported for the sake of readability.

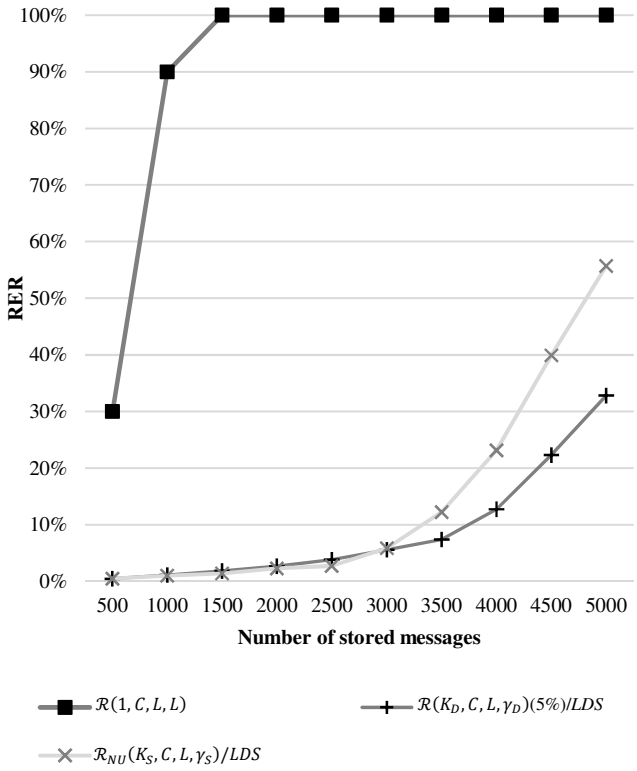



Fig. 16 The RER of the best CbNN models

It can be observed that the static approach ($\mathcal{R}_{NU}(K_S, C, L, \gamma_S)/LDS$) is able to provide excellent results with a RER of 0% (Δ±0%) up to 2000 stored messages. Then, the RER reaches 0,5% (Δ±0,03%) for 3000 messages, and the RER is 12,2% (Δ±1,3%) for 5000 messages.

The study of the RER of ($\mathcal{R}(K_D, C, L, \gamma_D)$ (5%)/LDS) shows that our dynamic clone allocation strategy has a very interesting message retrieving ratio, with the new erasure rate. It can be observed that for 5000 messages, the RER is 4,5% (Δ±0,06%). Moreover, the Δ values in this configuration is much more regular than in the other experiments: from 0.04% (for 1000 messages) to 0,06% (for 5000 message) in the figure. This regularity trend was not so clear in the first set of experiments performed with an erasure rate of 50%.

All these results show that dynamic CbNNs are able to store and retrieve large amounts of data more efficiently that their static counterparts. These results motivate us to further investigate dynamicity and to study how to dynamically compute a smartly tuned value of α. Our first experiments show

very interesting results, but still need some additional work to be published. Moreover, intuitively, considering all the sub-networks instead of the more recent used one, to store a new message should allow to increase memory capacity. This point will be investigated in our future work.

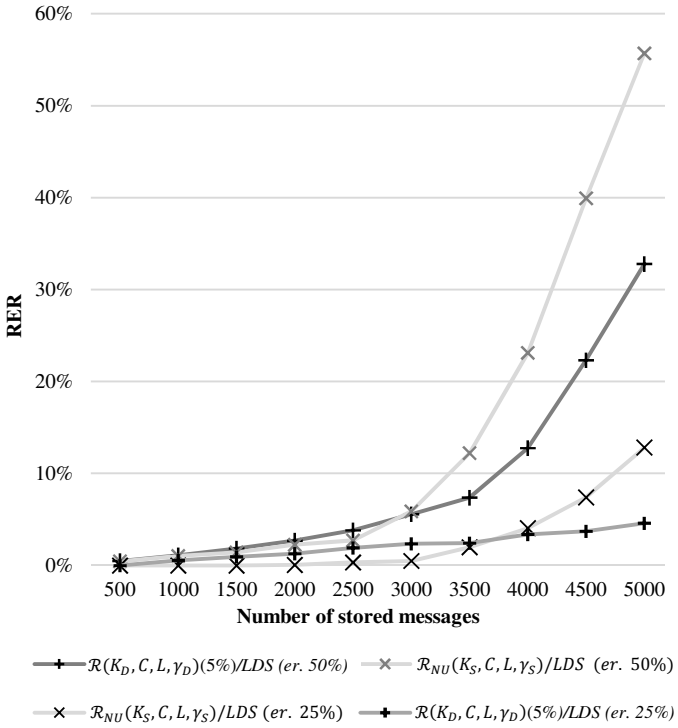



Fig. 17 The RER of static vs dynamic tuning with different erasure rates

## VIII. Conclusion

In this paper, we have introduced the concept of Clone-based Neural Network (CbNN) to consider realistic data (i.e., data non-uniformly distributed). Different strategies have been proposed to allocate clones and sub-networks in the CbNN. Experimental results show that all the proposed variants widely outperform the original ENN in terms of memory and recall abilities.

Considering an erasure rate of 50%, our results show that coupling non-uniform static allocation of clones with a dedicated storage algorithm LDS (CbNN variant $\mathcal{R}_{NU}(K_S, C, L, \gamma_S)$/LDS) is a more performant solution than the other static ones. $\mathcal{R}_{NU}(K_S, C, L, \gamma_S)$/LDS also competes with its dynamic counterpart $\mathcal{R}(K_D, C, L, \gamma_D)(5\%)$/LDS for small amounts of data. However, above a given threshold (~2500 data), dynamicity offers better performances.

When the erasure rate is reduced down-to 25% which corresponds to more classical and realistic stimuli, both static and dynamic CbNN variants allow to handle large amounts of data (i.e., the memory capacity is widely increased).

Hence, $\mathcal{R}_{NU}(K_S, C, L, \gamma_S)$/LDS is able to achieve a RER of 0% up to 2000 stored messages. Its dynamic counterpart $\mathcal{R}(K_D, C, L, \gamma_D)(5\%)$/LDS achieves a RER <1.9% for 2500 messages with an extreme regularity ($\Delta_{max}$= 0,06% for 5000 messages). Once again, results show that dynamicity offers better results and that the CbNN model allows the design of efficient associative memories.

Future work will be dedicated to dynamic tuning of the allocation threshold $\alpha_D$ to further improve the CbNN efficiency.

## IX. References


[1] G. Palm, “On associative memory,” *Biol. Cybern.*, vol. 36, no. 1, pp. 19–31, Feb. 1980.

[2] G. Palm, “Neural associative memories and sparse coding,” *Neural Netw.*, vol. 37, pp. 165–171, Jan. 2013.

[3] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[4] E. H. L. Aarts and J. H. M. Korst, “Boltzmann machines and their applications,” in *PARLE Parallel Architectures and Languages Europe*, J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, Eds. Springer Berlin Heidelberg, 1987, pp. 34–50.

[5] T. Kohonen, *Associative Memory: A System-Theoretical Approach*. Springer London, Limited, 2012.

[6] D. Willshaw, “Models of Distributed Associative Memory,” PhD thesis manuscript, University of Edinburgh, 1971.

[7] R. Hecht-Nielsen, “Neural Networks Letter: Cogent Confabulation,” *Neural Network*, vol. 18, no. 2, pp. 111–115, Mar. 2005.

[8] V. Gripon and C. Berrou, “Sparse Neural Networks With Large Learning Diversity,” *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1087–1096, Jul. 2011.

[9] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[10] L.K. Hansen and P. Salamon,"Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, issue 10, pp. 993 - 1001, 1990.

[11] J. Yang, et al, “Sparsely Connected Associative Memory Based on the Preferential Mechanism of Heuristic Annealed Topology”, *Journal of Shanghai Jiaotong Univ*, vol. 47, issue 07, pp.1009-1014, 2013.

[12] K. James et al, "Overcoming Catastrophic Forgetting in Neural Networks", *3rd ed., vol. 2.* National Academy of Sciences of the United States of America, Feb 13, 2017, pp.1-6.

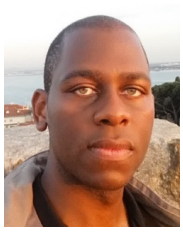
[13] K. JA. Graves et al., "Hybrid Computing using a Neural Network with Dynamic External Memory", *International Weekly Journal of Science*, Oct., pp 1-21, 2016.

[14] S.J. Wang, et al, “Sparse connection density unlies the maximal functional difference between random and scale-free networks,” The European Physical Journal B, vol. 86, issue 424, pp. 1-5, 2013.

[15] D. Holstein, et al, “Impact of noise and damage on collective dynamics of scale-free neuronal networks”, Physical Review E, vol. 87, issue 3, pp. 032717, 2013.H

[16] P. Sollich, et al, “Extensive parallel processing on scale-free networks,” Physical review letters, vol. 113, issue 23, pp. 238106, 2014.

[17] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks", *Science,* vol. 286, pp. 509-512, 1999.
[18] P. Kimani Mungai and R. Huang, " A Study on Merging Mechanisms of Simple Hopfield Network Models for Building Associative Memory", *IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC),* 2017.
[19] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006.
[20] B. Boguslawski, et al, "Huffman Coding for Storing Non-uniformly Distributed Messages in Networks of Neural Cliques," in *AAAI 2014: the 28th Conference on Artificial Intelligence*, Québec, Canada, 2014, vol. 1, pp. 262–268.
[21] H. Wouafo, C. Chavet, and P. Coussy, "Improving Storage of Patterns in Recurrent Neural Networks: Clones Based Model and Architecture," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Lisbon (Portugal), 2015.
[22] B. Boguslawski, et al, "Twin Neurons for Efficient Real-World Data Distribution in Networks of Neural Cliques: Applications in Power Management in Electronic Circuits," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 375–387, Feb. 2016.
[23] L. Lin, et al, "Organizing principles of real-time memory encoding: neural clique assemblies and universal neural codes," *Trends Neurosci.*, vol. 29, no. 1, pp. 48–57, Jan. 2006.
[24] V. Gripon and C. Berrou, "Nearly-optimal associative memories based on distributed constant weight codes," in *Information Theory and Applications Workshop (ITA), 2012*, 2012, pp. 269–273.
[25] P. Coussy, et al, "Fully Binary Neural Network Model and Optimized Hardware Architectures for Associative Memories," *J Emerg Technol Comput Syst*, vol. 11, no. 4, p. 35:1–35:23, avril 2015.
[26] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.
[27] H. Jarollahi, et al, "Reduced-complexity binary-weight-coded associative memories," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 2523–2527.



**Hugues WOUAFO** is a Post-Doc. at Lab-STICC (UMR CNRS) at the Université de Bretagne-Sud, France. He was graduated from the same university, (Ph.D. 2016, Master Degree on Computer Engineering 2012). He also works as a Consultant at Everygates (France) and KIRO'O (Cameroon). His research interests include system-level design and neuromorphic computing. IEEE member since 2012.

**Cyrille Chavet** received the M.S. and M.Ph. degrees in computer science from the Université Joseph Fourrier (Grenoble, France), in 2003, and the Ph.D. degree in computer engineering and computer science from the Université Bretagne-Sud (Lorient, France), in 2007. After 3 years in STMicroelectronics, Crolles, he held a post-doctoral position with the TIMA Laboratory, Grenoble, for a year. He is currently an Associate Professor with the Lab-STICC Laboratory (UMR CNRS). He is PC member and/or reviewer for conferences and journals (IEEE Trans. CAD, Trans. SP, DATE, ACM GLS-VLSI, ICASSP, ISCAS…), and he organized several events co-located with these conferences. He also published several papers in conferences and journals, and he is co-editor of a book: *Advanced Hardware Design for Error Correcting Codes* (Springer, 2015). His research interests are high level synthesis tools for hardware architectures, conflict-free memory mapping, advanced architecture for neural networks. He is IEEE senior member since 2005 and ACM SIGDA, member since 2006.

**Philippe Coussy** is a full professor in the Lab-STICC (UMR CNRS) at the Université de Bretagne-Sud, France, where he leads the "Communications Architectures Circuits and Systems (CACS)" department. He was graduated from Université Pierre et Marie Curie (MSc, 1999), Université de Bretagne-Sud (Ph.D., 2003 and Habilitation 2011). He is a member of the technical committee of the IEEE Signal Processing Society, Design and Implementation of Signal Processing Systems (DISPS) since 2011. He has organized several workshops and tutorials in many international conferences including DAC, DATE, CODES+ISSS and ASP-DAC. He was guest editor for several special issues of scientific journals and co-editor of two books (Springer). He regularly serves as a national and international scientific expert and participates as PC member in many international ACM/IEEE conferences and as reviewer for major IEEE/ACM journals. He is Associate Editor of the IEEE Signal Processing Letters for the design and implementation of signal processing systems. His research interests include system-level and computer-aided design, high-level synthesis and neuromorphic computing. He is IEEE senior member since 2001.