Storage Capacity of RAM-based Neural Networks: Pyramids

Paulo J. L. Adeodato [1] **John G. Taylor**

Centre for Neural Networks
King's College London, Mathematics Department,
The Strand WC2R 2LS, London, UK.

Abstract

Recently the authors developed a modular approach to assess the storage capacity of RAM-based neural networks [1] that can be applied to any architecture. It is based on *collisions of information* during the learning process. It has already been applied to the GNU architecture. In this paper, the technique is applied to the pyramid. The results explain practical problems reported in the literature and agree with experimental data and theoretical results obtained by architecture-specific techniques developed by other research groups. The successful application of that modular approach to the pyramid and GNU architectures — two out of the three most common ones — show it to be a useful tool for RAM-based neural networks specification for practical applications.

Keywords: RAM-based neural networks, storage capacity, information collision.

1 Introduction

Hardware implementability is one of the main features of RAM-based neural networks. They use Random Access Memory (RAM) chips as neurons. Another important feature is that the neurons can realise any function of their inputs. In most real life problems, however, this advantage of the RAM based neurons over the McCulloch-Pitts neuron is not transferred to the network because the dimension of the problem's feature space is much bigger

[1] Supported by The Brazilian Federal Agency for Postgraduate Studies (CAPES) grant No. 2296/91-8.
E-mail: P.adeodato@kcl.ac.uk

than the number of inputs per neuron — *fan-in* — of the chips currently available and each neuron sees only part of the whole feature space. This causes functionality reduction on the system compared to the single neuron as has been analysed by Al-Alawi & Stonham [2] in pyramidal architectures. Problem of similar nature occurs in hardware implementation of other classes of feedforward neural networks where different techniques have been applied in their analysis [4, 5].

Research on storage capacity of such neural networks has been done in pyramids [6] and in the general neural unit (GNU) configured as an autoassociative memory [7, 8]. Recently, Adeodato & Taylor [1] developed a general statistical approach to assess storage capacity of RAM-based neural networks based on collisions at the neuron level. This paper is a continuation of the previous one [1] with the aim of presenting how to combine the storage capacity of isolated neurons to assess those of the networks they form. It starts with a summary of the neuron model of collisions to assess storage capacity. Then the combination of the neurons' contributions is put forward for collisions in pyramids. Conclusions and future developments are discussed in the last section.

2 Collision in a neuron

The category of problems this paper is concerned with has binary input and binary output spaces. The neurons have permanent connections in the network. Input vectors (*input patterns*) address memory positions (*sites*) to store or retrieve (*train* or *recall*) the desired response (*target*). Training is supervised.

In training a neuron, a *collision* (or clash) of information is the storage of a target different from the previous ones stored for the same input pattern. Hence a collision destroys information previously stored in the addressed site. In the approach, the term *probability of collision* in a neuron refers to the probability of at least 1 collision occurring in at least 1 site of the neuron. It is a measure of the risk of disruption of information in the storage process. *Storage capacity* is the amount of patterns randomly drawn from a uniform distribution that can be stored in a neuron or network at a given risk of disruption of information — at a given probability of collision.

The following equations describe the model of collision in a single neuron which has been derived [1] using basic probability theory. They are the starting point of this paper. Some notation is introduced here:
$n \rightarrow$ neuron fan-in (number of inputs per neuron)
$s \rightarrow$ number of patterns presented to or stored in the neuron or network
$P(nc) \rightarrow$ probability of collision in a neuron
$P(\mathbf{x_i}) \rightarrow$ probability of having $\mathbf{x_i}$ as neuron input vector
$P(\overline{y}|\mathbf{x_i}) \rightarrow$ probability of having target $y = 0$ given the input vector $\mathbf{x_i}$
$P(\mathbf{x_i}|\overline{\mathbf{x}}_\mathbf{1}, ...\overline{\mathbf{x}}_\mathbf{i-1}) \rightarrow$ probability of having $\mathbf{x_i}$ as neuron input vector given the

input vectors are not $\mathbf{x_1}, ...\mathbf{x_{i-1}}$
$_A \rightarrow$ subscript to denote approximation by method A
$_B \rightarrow$ subscript to denote approximation by method B
$P(C_{pyr}) \rightarrow$ probability of collision in a pyramid
$P(C_l) \rightarrow$ probability of collision on a neuron in layer $\mathbf{l}$ of a pyramid
$m_l \rightarrow$ number of neurons in layer $\mathbf{l}$.

The probability of collision in a neuron for a given training data set based on its probability distribution is:

$$\begin{aligned} P(nc) &= 1 - \prod_{i=1}^{2^n} (P(\overline{\mathbf{x}}_\mathbf{i}|\overline{\mathbf{x}}_\mathbf{1}, ...\overline{\mathbf{x}}_\mathbf{i-1}) + P(\mathbf{x_i}|\overline{\mathbf{x}}_\mathbf{1}, ...\overline{\mathbf{x}}_\mathbf{i-1}) \\ & \quad (P(\overline{y}|\mathbf{x_i}) \cdot P(\overline{y}|\mathbf{x_i})^{(s-1)P(\mathbf{x_i})} + P(y|\mathbf{x_i}) \cdot P(y|\mathbf{x_i})^{(s-1)P(\mathbf{x_i})})) \end{aligned} \tag{1}$$

This is referred to as the exact model throughout the text. The analytical treatment of equation 1 is not simple and approximations were made. The probability distributions above are estimated from the relative frequencies on the training data set. Patterns randomly drawn from uniform distributions are used to assess the storage capacity of neural networks and, in this case, the probability of collision in a neuron becomes:

$$P(nc) = 1 - \prod_{i=1}^{2^n} (1 + \frac{1}{2^n - i + 1}((\frac{1}{2})^{(s-1)/2^n} - 1)) \tag{2}$$

2.1 Approximate models

In equation 2 for the exact model, the factors in the product are close to $\mathbf{1}$ except when $\mathbf{i}$ approaches 2^n. Approximation **A** is made considering it the product of a series of **1**s with the last factor ($i = 2^n$):

$$P_A(nc) = 1 - (\frac{1}{2})^{(s-1)/2^n} \tag{3}$$

Storage capacity is the inverse of this function, having the number of storable patterns as a function of the probability of collision and the neuron fan-in:

$$s_A = 1 - 2^n \log_2(1 - P(nc)) \tag{4}$$

Approximation **B** for the exact model considers that the conditional probabilities of addressing the sites do not interfere with each other, resulting in the equation below:

$$\begin{aligned} P_B(nc) &= 1 - (P(\overline{\mathbf{x}}_\mathbf{i}) + P(\mathbf{x_i}) \\ & \quad (P(\overline{y}|\mathbf{x_i}) \cdot P(\overline{y}|\mathbf{x_i})^{(s-1)P(\mathbf{x_i})} + P(y|\mathbf{x_i}) \cdot P(y|\mathbf{x_i})^{(s-1)P(\mathbf{x_i})}))^{2^n} \end{aligned} \tag{5}$$

It has the advantage of keeping the assessment of collisions dependent on the probability distribution of the training data set. Applied to patterns randomly drawn from a uniform distribution, the previous equation results in:

$$P_B(nc) = 1 - (1 + \frac{1}{2^n}((\frac{1}{2})^{(s-1)/2^n} - 1))^{2^n} \tag{6}$$

This equation is also inverted to give the storage capacity:

$$s_B = 1 - 2^n \log_2(2^n((1 - P(nc))^{2^{-n}} - 1) + 1) \tag{7}$$

A further approximation in equation 6 results in approximation **A**, supporting the validity of the approach-B. In fact, for small probabilities of collision, the approximations are similar and approximation B can be applied. Figures 1 and 2 show the curves of the model. Approximation A is a "delayed" version of the exact model. Approximation B follows approximation A for small levels of collision and tends to the limit $\underline{1 - e^{-1}\ (\approx 63\%)}$ when n and s increase. As there is no practical interest in probabilities of collision higher than 50%, approximation B becomes a powerful tool for assessing the storage capacity of a network for a specific training data set distribution.

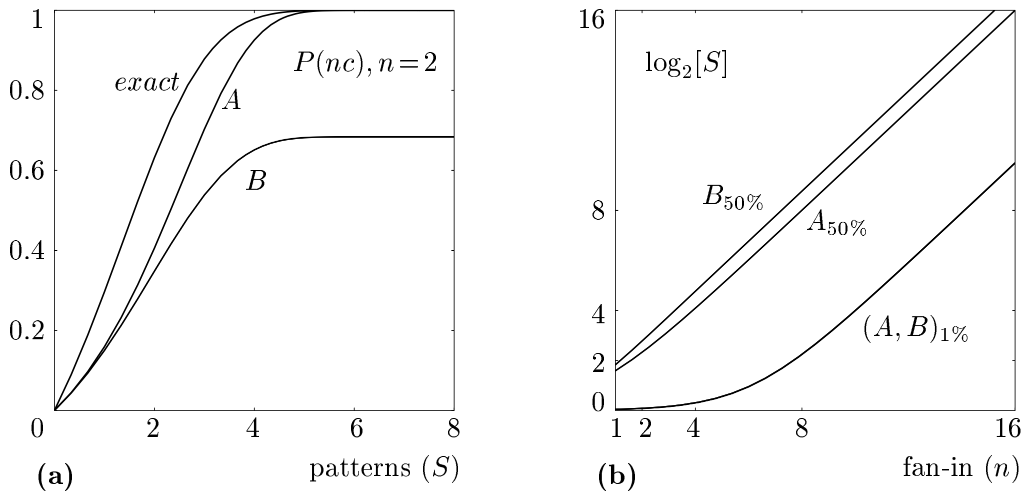

Figure 1: **(a)** Probability of collision as a function of the number of patterns for a single neuron with fan-in 2: exact model, approximation A and approximation B (equations 2, 3 and 6). **(b)** $\log_2$ of the storage capacity as a function of the fan-in for probabilities of collision of 50% and 1% (superimposed): approximations A and B (equations 4 and 7).

3 Collision in pyramids

The pyramid is assumed to be a tree of fixed connections composed of RAM-based neurons with fan-out of one arranged in layers [9]. The connection of the N input terminals of the pyramid to the N input features is assumed to be in a 1-to-1 fashion (no oversampling) so that the input space is divided in $\frac{N}{n}$ mutually exclusive sets of n features.

The crucial point to apply the modular model of neuron collision to pyramids is how to overcome the lack of information about the contents of the sites of the hidden layers' neurons. Efficient learning algorithms for RAM-based models [10, 11] tend to make the patterns associated with the target 1 have maximum Hamming distance from those associated with the target 0 in their hidden layer representations (internal representations). Since this is not enough for building up the model of collision for the architecture, the focus is turned to *optimal storage learning algorithms.* These algorithms have the power of storing all the patterns from a training data set without disruption (*e.g.* Back-Propagation Search Algorithm [3]), if at least one of the functions that satisfy the data set is implementable by the pyramid.

Assuming storage optimality, a collision is only propagated to a neuron in a higher layer if **all** the neurons from the previous layer which supply inputs to it have suffered a collision. A single neuron without collision can supply sufficient information to prevent the propagation of collisions to his successor in the next layer because it can divide the addressable sites of such neuron in two disjoint sets for the colliding targets — it only depends on the training algorithm being optimal. This interpretation is the basis for building up the model of collision in a pyramid. Collisions in neurons of the same layer are assumed to be independent events. Therefore the probability of collision — $P(C_l)$ — in a neuron of fan-in n_l of layer l is simply the product of the probabilities of collision in the neurons of the preceding layer which supply input to it. $P(nc_l)$ below is just the neuron model for collisions in any of its forms — exact, approximation-A or approximation-B. Mathematically:

$$\begin{array}{ll} P(C_1) = P(nc_1) & \\ P(C_l) = P(nc_{(l-1)})^{n_l} & \text{for} \quad l = 2, 3, ...L \end{array} \tag{8}$$

This is the building block for analysing pyramidal structures (tree structures, in general). The probability of collision in the whole pyramid — $P(C_{pyr})$ — is effect of neuron collisions propagated from the input to the output layer (L).

$$P(C_{pyr}) = P(nc_1) \cdot \prod_{l=2}^{L} P(C_l) \tag{9}$$

where the substitution of $P(C_l)$ results in:

$$P(C_{pyr}) = P(nc_1) \cdot \prod_{l=2}^{L} P(nc_{(l-1)})^{n_l} \tag{10}$$

In the common case of equal neurons in all the pyramid, the previous equation reduces to the one below.

$$P(C_{pyr}) = P(nc)^{n(L-1)+1} \tag{11}$$

This was the equation used to plot the figures in the recently published [1] paper. The plot reproduced below expresses storage capacity in terms of *probability of error-free storage* [6] which is simply the probability of having no collision in a pyramid. The curves are the application of the exact model to the networks from which Penny & Stonham [6] obtained their experimental data and show that this model fits well with their data.

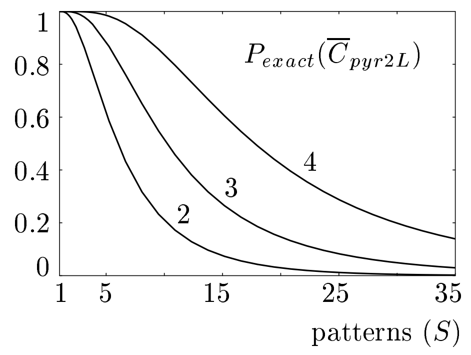

Figure 2: Probability of error-free storage [6] (exact model) as a function of the number of patterns in 2-layer pyramids with fan-ins 2, 3 and 4.

The function above (equation 11) is inverted to be composed with equations s_A or s_B for the approximate models of the neuron to give the storage capacity of the pyramid for a desired level of collision. This equation (below) is simply substituted in the storage capacity equations s_A and s_B:

$$P(nc) = 2^{\left(\frac{\log_2 P(C_{pyr})}{n(L-1)+1}\right)} \tag{12}$$

Having this technique available for assessing the storage capacity of pyramids, one can apply it to real problems in network design. Assume there is a classification task on an N-dimensional feature space where one wants to use a pyramid. Which fan-in should the neurons have in each layer of the pyramid? The figure below shows that there is a transition phase where the probability of error-free storage drops abruptly. This behaviour gives a good idea on how to make the choice of neuron fan-in depending on the amount of data one intends to store in the pyramid. The technique can also be applied to explain the problem of "saturation" reported by Filho *et. al.* [12].

Another application can be on how to choose different fan-in for neurons in each layer of the pyramid. Al-alawi & Stonham [2] have developed a technique based on pyramid functionality which could be used in such a choice. Using the technique presented here, the choice would be equivalent to theirs, as can be seen in the figure below, where approximation-A was applied to calculate the probability of error-free storage of five of the networks they had assessed. It should be emphasised that the curves do not cross for any probability of error-free storage.

In practical applications, cost has to be considered along with capacity in network design. The term *cost* here refers to the total amount of sites in a

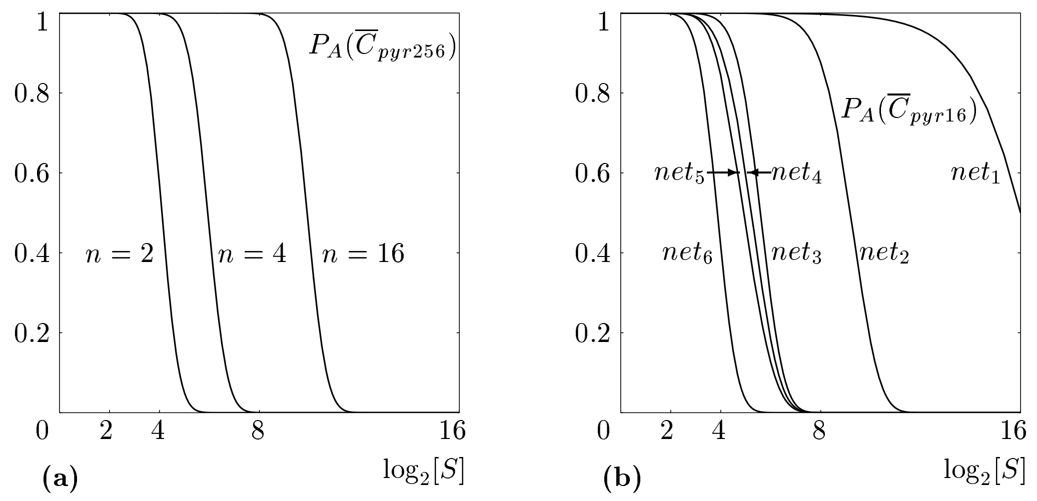

Figure 3: **(a)** Probability of error-free storage [6] as a function of the $\log_2$ of the number of patterns in a 256-dimensional space for pyramids with equal neurons (approximation A). Each pyramid has neurons of different fan-in — 2, 4 and 16). **(a)** Probability of error-free storage [6] as a function of the $\log_2$ of the number of patterns in a 16-dimensional space for pyramids with different neurons (approximation A). This refers to the pyramids presented by Al-alawi & Stonham [2].

network — in all its neurons. Assuming equal neurons in each layer (l) with fan-in n_l, the total number of neurons per layer (m_l) is:

$$\begin{aligned} & m_L = 1 \\ & m_l = \textstyle\prod_{j=l+1}^{L} n_j \qquad\qquad \text{for} \quad l = 1, 2, ...L-1 \end{aligned} \tag{13}$$

The cost of a layer is just the product of the cost of a neuron (2^{n_l}) and the number of neurons in that layer (m_l). The cost of the pyramid is then the sum of the costs of all layers:

$$cost(pyr) = \sum_{l=1}^{L} 2^{n_l} \cdot m_l \tag{14}$$

For equal neurons with fan-in n in all the pyramid, the equation above reduces to:

$$cost(pyr) = 2^n \sum_{l=1}^{L} n^{l-1} \tag{15}$$

The figure below presents the ratio the logarithm base-2 of the storage capacity and the logarithm base-2 of the cost as a function of the level of collision for several fan-ins for the same N-dimensional space. It is clear that the reduction in storage capacity is bigger than the reduction in cost when one reduces the fan-in of the neurons. This "extra" reduction is caused by the multiplicity in the representation of the functions, as has been explained by Al-alawi & Stonham [2]. This is another result coherent with previous architecture-specific techniques based on other approaches.

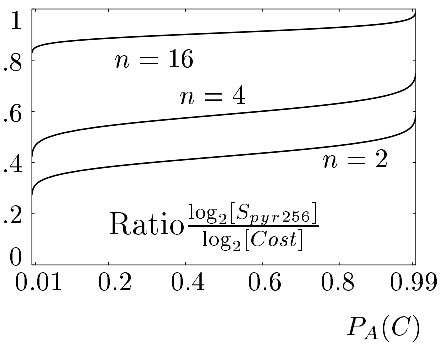

Figure 4: Ratio of $\log_2$ of the storage capacity over $\log_2$ of the cost as a function of the probability of collision in 256-pyramids (approximation A). Each pyramid has equal neurons, with fan-ins of 2, 4 and 16).

4 Conclusions

This paper presented a technique to model the storage capacity of RAM-based pyramids based on the probabilistic model developed [1] to assess storage capacity of isolated neurons. The definition of the concept of optimal storage learning algorithm was crucial to overcome the problem of assessing the effect of collisions in the hidden layers. The theory explains problems reported in specific systems [12] and the theoretical and experimental results agree with the ones obtained by other techniques [2, 6] for pyramid analysis. For the GNU architecture as an autoassociative memory, the model [1] has already been developed and the theoretical and experimental results also agreed with the ones obtained by other techniques [7, 8]. The combination of both techniques to analyse the use of pyramids as nodes of the GNU architecture shows an increase in the storage capacity and a better storage capacity/cost ratio compared to the single neuron node (results not presented here).

There are important features in the approach developed by the authors. First, it is simple and modular as shown in this paper. Second, it is broadly applicable, fitting well with the data from a broad range — pyramids and GNUs. It is the most general tool developed for storage capacity assessment of RAM-based neural networks so far. Third, it is data-dependent; a very promising feature. This could be used to define not only the connections of RAM-based networks to the feature space but also its inner construction in a "smart" fashion as part of a global network optimisation technique.

When retrievability is analysed, both the neuron model and its learning algorithm are crucial to assess the performance of the network. In the model here, no assumption was made on the neuron apart from it being RAM-based. The algorithm, however, was assumed to be optimal in the storage capacity. This assumption is very clearly defined for tree-like architectures (including pyramids) but difficult to define in structures where there are neurons with fan-out bigger than 1 (*e.g.* the trapezium) which are subjected to inherently competing information.

Further theoretical work needs to be carried out to relate and combine this approach with the ones from computational learning theory and information theory. At the moment, research is being done to apply this technique to other RAM-based architectures where collisions occur (not the N-tuple classifiers). Also, a collision-detector simulator based on Al-alawi's back-propagation search algorithm [3] is being produced to assess experimentally the storage capacity of RAM-based pyramids larger than the ones for which data is available [6].

References

[1] Adeodato P. J. L., Taylor J. G.: Analysis of the storage capacity of RAM-based neural networks. In: *Proc. of Weightless Neural Networks Workshop*, D. L. Bisset (ed.), Canterbury, UK, Sep. 1995, 103–110.

[2] Al-Alawi R., Stonham T. J.: Functionality of multilayer Boolean neural networks. *Electronics Letters*, **25**, 1989, 657–659.

[3] Al-Alawi R., Stonham T. J.: A training strategy and functionality analysis of digital multi-layer neural networks. *Journal of Intelligent Systems*, **2**, 1989, 53–93.

[4] Shawe-Taylor J. S., Anthony M. H., Kern W.: Classes of feedforward neural networks and their circuit complexity. *Neural Networks*, **5**, 1992, 971–977.

[5] Beiu V., Taylor J. G.: On the circuit complexity of sigmoid feedforward neural networks. *Neural Networks*, (accepted for publication).

[6] Penny W., Stonham T. J.: Storage capacity of multilayer Boolean neural networks. *Electronics Letters*, **29**, 1993, 1340–1341.

[7] Wong, K. Y. M., Sherrington D.: Storage properties of randomly connected Boolean neural networks for associative memory. *Europhysics Letters*, **7**, 1988, 197–202.

[8] Braga A. P.: Predicting contradictions in the storage process of diluted recurrent Boolean neural networks. *Electronics Letters*, **30**, 1994, 55–56.

[9] Aleksander I.: Neural systems engineering: towards a unified design discipline? *IEE Comp. & Cont. Eng. Jour.*, **1**, 1990, 259–265.

[10] Bowmaker R. G., Coghill G. G.: Improved recognition capabilities of goal seeking neurons. *Electronics Letters*, **28**, 1992, 220–221.

[11] Guan Y., Clarkson T., Taylor J. G., Gorse D.: Noisy reinforcement training for pRAM nets. *Neural Networks*, **7**, 1994, 523–538.

[12] Filho E. C. D. B. C., Fairhurst M. C., Bisset D. L.: Analysis of saturation problem in RAM-based neural networks. *Electronics Letters*, **28**, 1992, 345–347.