Fat-Fast VG-RAM WNN: A high performance approach

Article *in* Neurocomputing · December 2015

Impact Factor: 2.08 · DOI: 10.1016/j.neucom.2015.06.104

7 authors, including:

Avelino Forechi

Universidade Federal do Espírito Santo

8 PUBLICATIONS **10** CITATIONS

SEE PROFILE

Alberto F. De Souza

Universidade Federal do Espírito Santo

84 PUBLICATIONS **218** CITATIONS

SEE PROFILE

Jorcy de Oliveira Neto

Universidade Federal do Espírito Santo

11 PUBLICATIONS **10** CITATIONS

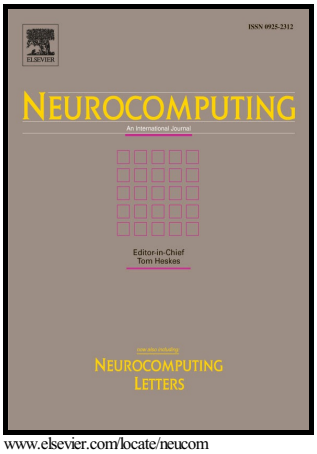
SEE PROFILE

Claudine Badue

Universidade Federal do Espírito Santo

31 PUBLICATIONS **211** CITATIONS

SEE PROFILE



www.elsevier.com/locate/neucom

Fat-Fast VG-RAM WNN: A High Performance Approach

Avelino Forechi, Alberto F. De Souza, Jorcy de Oliveira Neto, Edilson de Aguiar, Claudine Badue, Artur Garcez, Oliveira-Santos Thiago

PII: S0925-2312(15)01987-6
DOI: http://dx.doi.org/10.1016/j.neucom.2015.06.104
Reference: NEUCOM16564

To appear in: *Neurocomputing*

Received date: 7 April 2014
Revised date: 2 May 2015
Accepted date: 20 June 2015

Cite this article as: Avelino Forechi, Alberto F. De Souza, Jorcy de Oliveira Neto, Edilson de Aguiar, Claudine Badue, Artur Garcez and Oliveira-Santos Thiago, Fat-Fast VG-RAM WNN: A High Performance Approach, *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2015.06.104

Fat-Fast VG-RAM WNN: A High Performance Approach

Avelino Forechi[ac1], Alberto F. De Souza[a], Jorcy de Oliveira Neto[a], Edilson de Aguiar[b], Claudine Badue[a], Artur Garcez[c], Thiago Oliveira-Santos[a]

[a] Departamento de Informática, Universidade Federal do Espírito Santo
Av. Fernando Ferrari 541, 29075-910, Vitória - ES, Brazil
[b] Departamento de Computação e Eletrônica, Universidade Federal do Espírito Santo
Rodovia BR 101 Norte Km. 60, 29932-540, São Mateus - ES, Brazil
[c] Department of Computer Science, City University London
Northampton Square, London, EC1V 0HB, UK

Abstract

The Virtual Generalizing Random Access Memory Weightless Neural Network (VG-RAM WNN) is a type of WNN that only requires storage capacity proportional to the training set. As such, it is an effective machine learning technique that offers simple implementation and fast training – it can be made in one shot. However, the VG-RAM WNN test time for applications that require many training samples can be large, since it increases with the size of the memory of each neuron. In this paper, we present Fat-Fast VG-RAM WNNs. Fat-Fast VG-RAM WNNs employ multi-index chained hashing for fast neuron memory search. Our chained hashing technique increases the VG-RAM memory consumption (fat) but reduces test time substantially (fast), while keeping most of its machine learning performance. To address the memory consumption problem, we employ a data clustering technique to reduce the overall size of the neurons' memory. This can be achieved by replacing clusters of neurons' memory by their respective centroid values. With our approach, we were able to reduce VG-RAM WNN test time and memory footprint, while maintaining a high and acceptable machine learning performance. We performed experiments with the Fat-Fast VG-RAM WNN applied to two recognition problems: (i) handwritten digit recognition, and (ii) traffic sign recognition. Our experimental results showed that, in both recognition problems, our new VG-RAM WNN approach was able to run three orders of magnitude faster and consume two orders of magnitude less memory than standard VG-RAM, while experiencing only a small reduction in recognition performance.

Keywords: WNN, VG-RAM, multi-index chained hashing, data clustering, traffic sign recognition, handwritten digit recognition.

[1] Corresponding author.
Email addresses: avelino@lcad.inf.ufes.br (Avelino Forechi), alberto@lcad.inf.ufes.br (Alberto F. De Souza), jorcyd@lcad.inf.ufes.br (Jorcy de Oliveira Neto), edilson.de.aguiar@gmail.com (Edilson de Aguiar), claudine@lcad.inf.ufes.br (Claudine Badue), a.garcez@city.ac.uk (Artur Garcez), todsantos@inf.ufes.br (Thiago Oliveira-Santos).

1. Introduction

Weightless Neural Networks (WNNs [1]), also called n-tuple classifiers [2], are powerful machine learning techniques [3] that offer fast training and test. In contrast with standard feed-forward Neural Networks [4], which store knowledge in the form of synaptic weights, WNN store knowledge mainly in Random Access Memory (RAM) inside the network's neurons, which makes training and test fast. Their first version was proposed by Bledsoe and Browning in 1959 [5] and, after that, many other implementations were proposed (see [1] for a review). All implementations share the following two properties: (i) the synapses of WNN neurons do not carry weights (hence the denomination of these neural networks), and each of them only collect a single bit (zero or one) from the network's inputs; and (ii) the network's knowledge is stored in binary look-up tables inside the network's neurons. It is important to note that the interconnection pattern between a layer of WNN neurons and its inputs stores knowledge as well, but typically a static knowledge, i.e. once created, the interconnection pattern does not change.

In WNNs, the synapses of each neuron of a layer of neurons collect a vector of bits from the network's input (or other neural layers of the network) and use this vector of bits as a key to access the neuron's look-up table. During training, this access key is used to store an expected output, associated with the input vector, into the look-up table (supervised learning). During test, the input vector is used as an access key for reading one of the previously learned outputs from the look-up table. In its most simple implementation, the look-up table is a random access memory (RAM); therefore, training and test can be made in one shot. Training basically consists of storing the desired output value into the look-up table's address that is associated with the neuron input vector, while test consists of retrieving the information stored in the look-up table's address that is associated with the neuron input vector. However, in this simple implementation, the neuron memory size may become prohibitive if the neuron input vector is too large (the neuron memory size for this simple WNN is equal 2^p, where p is the neuron input vector size).

To overcome this limitation, many solutions have been investigated. A well-known solution is the Wilkes, Stonham and Aleksander Recognition Device (WiSARD), which in algorithm form is identical to the n-tuple recognition algorithm of Bledsoe and Browning [5]. WiSARD overcomes this limitation by dividing the p-sized neuron input vector (collected by its synapses) in q segments. Each segment is used to address a specific RAM memory module of size $2^{p/q}$ – each one of these modules is responsible for recognizing a single class of input pattern of interest (see [6] for a detailed description of WiSARD and [7] for a shorter one). With this organization, the total amount of memory required per neuron is reduced from 2^p to $q \times 2^{p/q}$. This framework, and many variants, have been used for a large variety of applications, including indoor positioning systems [8], robot localization system [9, 10, 11], recognition of DNA chains [12], feature tracking in images [13], data clustering [14], and audio recognition [15]. However, given the constraints imposed by dividing the neuron input vector in q segments, the representation capacity of the network is reduced [16], which hinders its generalization ability [17]. As a result, the performance of applications using WiSARD may be lower than that obtained with related techniques, such as Virtual Generalizing Random Access Memory WNN (VG-RAM WNN, or VG-RAM for short). This can be seen in [18], where the accuracy of a traffic sign recognition system with WiSARD and VG-RAM were compared.

The VG-RAM WNN is a type of WNN that only requires storage capacity proportional to the training set [7]. They have shown high classification performance for a variety of multi-class classification applications, including text categorization [19, 20], face recognition [21, 22, 23], and traffic sign detection and recognition [18, 24, 25]. During training, VG-RAM neurons store pairs of associated input-output patterns, i.e. input binary vectors and associated outputs, instead of only the output, as mentioned above for the WNNs. Thanks to their training method, VG-RAM's memory footprint is optimized, and its training performance in terms of time is maintained. One disadvantage of VG-RAMs, however, is their test time, which depends on the size of their neurons' memory, i.e. the number of trained input-output pairs. During test, VG-RAM's neurons compute their outputs by searching for the learned input-output pair whose input is the closest to the current neuron's input – the output of this input-output pair is used as the neuron's output. The search for the closest pair is sequential, which is costly in terms of time if there are many training samples.

To cope with this problem, in this paper we present a Fat-Fast version of VG-RAM WNNs. Our Fat-Fast VG-RAMs employ multi-index chained hashing for fast searching in the neuron's memory. Even though our chained hashing technique increases the VG-RAM memory's consumption (fat), it reduces test time substantially (fast), while keeping most of the machine learning performance of VG-RAM. To address the increase in memory consumption of the Fat-Fast VG-RAMs, we have employed data clustering techniques for reducing the overall size of their neurons' memory (a preliminary study of memory clustering techniques for VG-RAM is presented in [26]). With this approach, we were able to reduce the VG-RAM test time and memory footprint, while maintaining a high and acceptable machine learning performance. To validate the Fat-Fast VG-RAM, we performed experiments with two machine learning problems: handwritten digit recognition and traffic sign recognition. Our experimental results showed that, in both problems, our new VG-RAM approach was able to run three orders of magnitude faster and consume two orders of magnitude less memory than standard VG-RAM, while experiencing only a small reduction in classification performance.

This paper is organized as follows. After this introduction, in Section 2, we describe the VG-RAM WNN and the techniques employed to improve VG-RAM's test time and to alleviate memory consumption. In Section 3 we describe the experimental methodology, in Section 4 our experimental results and, in Section 5, we present an overall discussion of this work. Finally, we conclude and point directions for future work in Section 6.

2. Framework

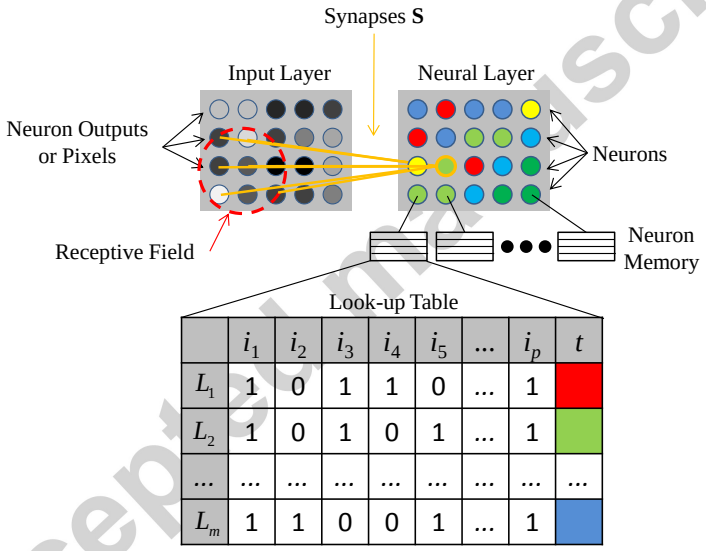
Section 2.1 presents our VG-RAM's architecture designed to solve image-related classification problems by employing a committee of VG-RAM units, called neurons, which are individually responsible to represent the binary patterns extracted from the input and their corresponding label. Those neurons are organized in layers and can also serve as input to further layers in the architecture. As a machine learning method, its use consists of a training phase to store pairs of inputs and labels, which will be compared to new inputs in the testing phase. Considering that its testing-time increases linearly with the number of training samples, it is proposed in Section 2.2 a faster neuron memory search leveraged by an indexed data structure with sub-linear runtime for uniformly distributed patterns. Unfortunately, this data structure requires a significant amount of additional memory to index the memory of each VG-RAM neuron. Thus, to

compensate this memory overhead, it is presented in Section 2.3 a clustering method over the VG-RAM neuron's memory to compress the number of representative input patterns associated to each label in each neuron memory.

2.1 VG-RAM

The architecture of VG-RAM WNNs comprises neuron layers with many neurons connected to Input Layers (e.g. images or other neuron layers) through a set $S = \{s_1, \ldots, s_p\}$ of synapses. Since VG-RAM neurons generate outputs in the form of label values, t, the output of a neural layer can also function as an image, where each pixel is the output of a neuron.

The synapses of a VG-RAM neuron are responsible for sampling a binary vector $I = \{i_1, \ldots, i_p\}$ (one bit per synapse) from the input layer according to the neuron's Receptive Field. Each bit of vector I is computed using a synapse mapping function that transforms non-binary values in binary values. Individual neurons have private memories, i.e. look-up tables, that store sets $L = \{L_1, \ldots, L_j, \ldots, L_m\}$ of learned binary input vectors, I, and corresponding output labels, t, i.e. $L_j = (I_j, t_j)$ (input-output pairs). An illustration of a single layer of such a network is presented in Figure 1.



	i_1	i_2	i_3	i_4	i_5	...	i_p	t
L_1	1	0	1	1	0	...	1	
L_2	1	0	1	0	1	...	1	
...	...	...	...	...	...	...	...	...
L_m	1	1	0	0	1	...	1	

Figure 1: Illustration of a single layer VG-RAM WNN. The VG-RAM WNN Neural Layer comprises many neurons that are connected to the Input Layer (e.g. an image) by a set of Synapses $S = \{s_1, \ldots, s_p\}$. These synapses sample binary input vectors $I = \{i_1, \ldots, i_p\}$ from the Input Layer according to the Receptive Field of each neuron. The bits of these vectors are computed using a synapse mapping function that transforms non-binary values in binary values. Each neuron of the Neural Layer has a private memory (Look-up Table) of size m that stores a set $L = \{L_1, \ldots, L_j, \ldots, L_m\}$ of $L_j = (I_j, t_j)$ input-output pairs learned during training. Each neuron shows an output activation value, t, read from its memory (Look-up Table column t and Neural Layer colored circles). This value corresponds to the output of the input-output pair, j, whose input I_j is the closest to the current binary input vector I extracted by the neuron's synapses during testing phase.

VG-RAMs operate in two phases: a training phase, in which neurons learn new pairs of binary input vectors and corresponding output labels (input-output pairs); and a testing phase, in which neurons receive binary input vectors and respond with the label values associated with the closest binary input vectors in the input-output pairs previously learned.

More specifically, in the training phase, each neuron includes a new input-output pair, or line L, in its local memory as follows. Firstly, an input image is set in the Input Layer of the VG-RAM (Figure 1). Secondly, the corresponding Neuron Outputs are set in the Neural Layer of the VG-RAM (expected output value t of each neuron for the input image set). Finally, one input-output pair (binary input vector I and corresponding output label t) is extracted for each neuron and is added into its memory as a new line L = (I, t). In the test phase, each neuron computes an output label t as follows. Firstly, an input image is set in the Input Layer of the VG-RAM. Secondly, a binary input vector I is extracted for each neuron. Finally, each neuron searches its memory to find the input-output pair $L_j = (I_j, t_j)$ whose input I_j is the closest to the extracted input I, and sets the corresponding Neuron Output of the Neural Layer with the output value t_j of this pair. In case of more than one pair with an input at the same minimum distance of the extracted input, the output value is randomly chosen among them. The memory search is sequential and the distance function is the Hamming distance.

The Hamming distance between two binary patterns can be efficiently computed at machine code level in current 64-bit CPUs and GPUs of personal computers using two instructions: one instruction to identify the bits that differ in the two binary patterns, i.e. bit-wise exclusive-or; and another instruction to count these bits, i.e. population count instruction.

VG-RAM WNNs scores high machine learning performance within an acceptable response time for many daily-life applications with (i) not a very large number of training examples or (ii) not requiring real-time performance. However, lately, the amount of training data has grown together with the need for real-time machine learning applications. Given the architecture of this type of network, two major limitations arise when dealing with applications with large training datasets: test time and memory usage. The first is associated with the search for the closest pattern, which is performed sequentially in the memory of each neuron. The second is associated with the pairs of examples that have to be kept in memory. These hinder not only the wide use of VG-RAM with real time applications, but also its use with devices with limited amount of memory, such as low-power embedded systems [26]. Both problems are related to the size of L, which may slow-down the sequential search and increase the memory usage. The size of L is proportional to the number of synapses p and the number of samples m, where the latter is more likely to grow in future relevant applications.

2.2 Faster Neuron Memory Search with Fat-Fast VG-RAM

The search for the nearest input-output pair $L_j = (I_j, t_j)$ whose input I_j is the closest to the neuron input vector I is a k-nearest neighbor search in a hamming space, where k is equal to 1. To efficiently address this problem at bit level and return the exact k nearest neighbors, Narouzi et. al [27, 28] employed a method where the searched pattern I is divided in sub-patterns that, in turn, are used to index multiple hash tables (one table per sub-pattern). They have shown that a good compromise between access speed and memory usage can be found for the hash tables. Following in the same direction, this paper brings the multiple hash tables idea into the VGRAM WNN in order to speed up the test phase. Differently from the approach mentioned above, we are not restricted to an exact match of the nearest neighbor. Instead, we are interested on a good trade-off between classification accuracy and test time, when considering real world applications.

In our approach, multiple hash tables are created for the memory of each VG-RAM neuron considering the binary space defined by its input vector I; we named these

modified VG-RAM neurons Fat-Fast VG-RAM neurons. Figure 2 gives an overview of our multi-index hash approach to the neuron memory architecture.

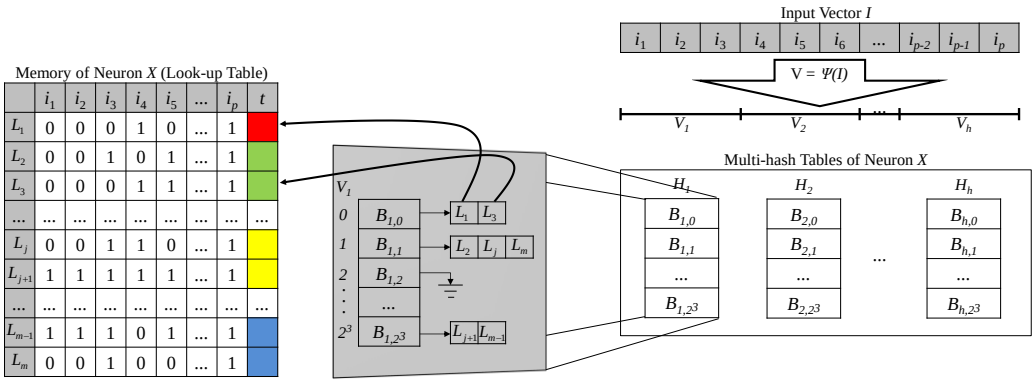


Figure 2: Fat-Fast VG-RAM neuron memory. A Fat-Fast Neuron X reads, via its synapses, the binary input vector I. The figure illustrates the process of mapping an input vector, I, of size equal to p bits, into buckets, $B_{k,v}$, that stores references to the set of memory lines that have to be inspected during memory search (see arrows above). The input vector I is firstly divided into sub-vectors, $V = \{V_1, \ldots, V_k \ldots, V_h\}$, where $V_k = \{0, 1, \ldots, v, \ldots, 2^{p/h}\}$. These sub-vectors are used to address the multi-index hash table, $H = \{H_1, \ldots H_k \ldots, H_h\}$. Each address, V_k, points to a bucket $B_{k,v}$ of lines, L, that should be considered in the search. Note that this illustration uses binary codes, V_k, of 3 bits only for clarity.

Since we are working with bits, a very effective hash function $\Psi(I)$ can be created by simply dividing the binary input vector $I = \{i_1, \ldots, i_p\}$ into h binary segments, i.e. binary sub-vectors $V = \{V_1, \ldots, V_k \ldots, V_h\}$. Each binary sub-vector comprises p/h bits regularly sampled from I, i.e. $V_k = \{0, 1, \ldots, v, \ldots, 2^{p/h}\}$. V is used as index to the multi-index hash table $H = \{H_1, \ldots H_k \ldots, H_h\}$, where $1 \leq h \leq p$. Each V_k is directly used as an address of the respective hash table, H_k, of H. Each hash table H_k has $2^{p/h}$ unique buckets, $B_{k,v}$, containing $|B_{k,v}|$ indexes to memory lines. Buckets are necessary because several learned memory lines may share the same sub-vector code value, i.e., may have conflicting V_k addresses (e.g., see conflicting lines L_1 and L_3 of bucket $B_{1,0}$ in Figure 2). Buckets can store references to up to m memory lines that have conflicting addresses, where m is the number of learned input-output pairs, i.e. |L|. An efficient implementation of buckets can be achieved with arrays of variable size.

To train a Fat-Fast neuron, it is necessary to fill its memory with input-output pairs $L_j = (I_j, t_j)$ following the same training procedure of a standard VG-RAM. In addition, it is necessary to populate the multi-index hash table, H, with references to each one of the learned pairs. For that, the binary codes V are extracted from I_j using $\Psi(I_j)$, and one reference to L_j is added to the respective bucket of each hash table of H.

During test, fast search for the nearest neighbor of a binary input vector I can be performed for each Fat-Fast VG-RAM neuron as follows. Firstly, the function $\Psi(I)$ is used to retrieve V. Subsequently, each V_k is used as an address to the bucket $B_{k,v}$. Finally, the input vector I_j of each memory line $L_j = (I_j, t_j)$ of each selected bucket is examined as a candidate for the nearest input to the neuron input vector I. Note that, all candidates are compared with I using the Hamming distance between the full bit vectors, i.e. the p bits of I and I_j. The Fat-Fast VG-RAM output is the label t_j of the nearest line, L_j. If more than one candidate line is at the same closest distance to I, the neuron output is randomly selected among them.

It may occur that the number of lines in a selected bucket is zero. In these cases, no memory line is considered for that bucket. If none of the h selected buckets has candidate lines, the neuron output is taken from a random line of memory. On the other

hand, it may also occur that there are too many lines in one or more selected buckets. In such cases, the searching time could increase substantially. Hence, in these cases, a maximum of r candidates in each large bucket (bucket size bigger than r) is randomly selected for comparison with the input vector I, where r is a Fat-Fast parameter.

Indeed, the parameter r is required to be tuned manually and accordingly to each particular application, i.e., one needs to balance the trade-off between accuracy and runtime, conditioned to the amount of available RAM. As to be discussed, a higher parameter r increases the accuracy close to VG-RAM levels, but at the same time slows down the system, and vice versa. The multi-index hash tables of Fat-Fast VG-RAM WNN neurons are only used as a mean to reduce the number of memory lines to be examined. In practice, converting the vector I into V can be costly if performed inappropriately. In order to avoid unnecessary processing, in our Fat-Fast implementations, we kept p/h equal to the number of bits used in the basic types of common programming languages, which facilitates retrieving the address V_k of each bucket. Yet, it is possible to determine an optimal p/h ratio that minimizes the total number of collisions inside the multi-index hash [27, 28]. Thus, given a fixed number of synapses p, the optimal value for h is $\log_2(m)$, where m is the total number of training examples. But here we consider p/h fixed because, otherwise, it could affect the runtime measurements given that h is computed based on different values of m.

The search time of each neuron in standard VGRAM WNN is O(m) because the hamming distance has to be calculated for every example in the memory. The search time with the multi-index hash tables approach is at the most O(h×r) because only up to r lines per hash table will have the hamming distance computed. For applications with large number of training examples, the gain might be huge for small r because h << m. Since we are interested in the overall machine learning power of the network, it is important to know the effect of changes in r. The larger the value of r is, the more the search for the nearest neighbor approximates the exact search. In the other hand, the smaller the value of r is, the more is the impact on machine learning performance, since some good neuron memory lines could be left out of the hamming distance comparison.

2.3 Memory Size Reduction

Our Fat-Fast VGRAM WNN re-organizes the neurons' memory using a special multi-index hash system that increases the network's memory footprint in exchange for faster test time. To tackle the memory size increase issue, we propose the use of data clustering techniques to reduce the overall size of the neurons' memory [26]. Assuming that the memory of each neuron has a large amount of redundant or irrelevant information, we should be able to accomplish memory reduction by carefully eliminating input-output pairs of the neurons' memory, L. This is achieved by using a clustering algorithm to find the most relevant input-output pairs in the memory of each neuron.

From a number of clustering techniques [29, 30, 31, 32], and considering the fact that the clustering procedure needs to be done many times for each neuron (see below), we decided to use k-Means [32] due to its simplicity and efficiency. The k-Means algorithm partitions a set I of |I| vectors into k clusters, k < |I|, where the parameter k is set a priori. This iterative partitioning scheme minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. For our experiments, we used the Hamming distance to define the centroids of each cluster.

Our memory reduction framework is illustrated in Figure 3. Initially, the original memory of each neuron, containing m lines, is sorted according to its output labels t (i.e.

same handwritten digit or traffic sign of same type). The result is a set of lines ordered according to each output type, e.g. the set of lines $L^1 = \{L^1_1, \ldots, L^1_{m_1}\}$ for outputs of type equal to label t = 1, $L^2 = \{L^2_1, \ldots, L^2_{m_2}\}$ for outputs of type equal to label t = 2, until $L^l = \{L^l_1, \ldots, L^l_{m_l}\}$ for outputs of type equal to label t = l. Subsequently, we apply k-Means separately to each set of lines L^1, L^2, …, L^l, partitioning each one of them into k clusters. Each cluster has a centroid C that is equivalent to a memory line L. k-Means computes the set $C^1 = \{C^1_1, \ldots, C^1_{k_1}\}$ of centroids from the lines of set L^1, where $k_1 < m_1$ and $m_1 = |L^1|$; the set $C^2 = \{C^2_1, \ldots, C^2_{k2}\}$ of centroids from the lines of set L^2, where $k_2 < m_2$; and so on until the set $C^l = \{C^l_1, \ldots, C^l_{k_l}\}$, where $k_l < m_l$. The centroid sets C^1, C^2, …, C^l become the new memory entries for the respective neuron (see Figure 3). This process is repeated for all neurons, i.e. the clustering process is performed n × l times. As a result, the number of entries in the network memory is reduced according to the specified number of clusters, k_j, of each label type j. To maintain the original memory balance between entries for the different output types, we use different values of k for each label type, i.e. $k_j = m_j \times rl$, where $0 < rl < 1$ is the memory retention level – the lower the rl, the higher the memory reduction (rl expresses the amount of memory that remains after memory reduction). As a general rule, the parameter rl can be chosen as closest to one as possible, regarding the amount of available memory. We believe that this formulation, which determines the number of clusters k, is an important property of our approach since it allows changing the memory compression level depending on the application requirements. For instance, this approach could be used for resource-restricted applications by appropriately selecting the parameter rl that best fit the hardware requirements. For further discussion about tuning the parameter rl the interested reader can refer to [26], where the parameter rl is studied under a more resource-restricted settings.

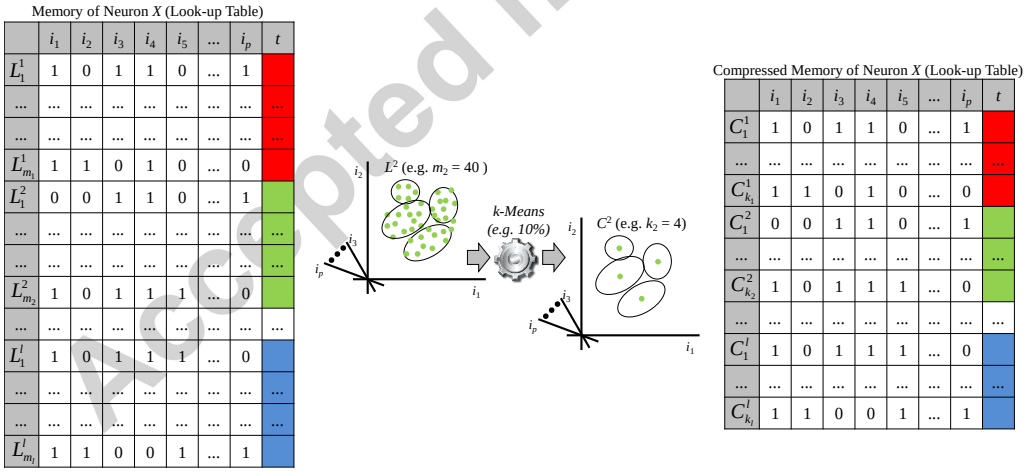


Figure 3: Memory reduction framework. The original memory of each neuron, containing m lines, is treated as a set of points in a multi-dimensional space. First, the neuron's memory is grouped according to its output type t. Thereafter, each group is clustered separately using k-Means, where k is defined based on the trade-off between accuracy x runtime. At the end, the resulting centroids for each output type group become the new entries for the reduced memory, substituting the original memory information.

It is important to note that our approach changes the contents of the neurons' memory. It can even create a new entry that was not part of the learning process. We see this as an advantage of our framework since it is able to summarize the memory information according to the data. Please note that the input patterns $I_j = \{i_1, \ldots, i_p\}$ of

all centroids $C_j = (I_j, t_j)$ must be binary, therefore, the centroids' elements need to be rounded to values 0 or 1.

3. Experimental Methodology

We implemented two recognition systems for evaluating the Fat-Fast VG-RAM WNN: (i) a handwritten digit recognition system, and (ii) a traffic sign recognition system. We performed experiments with both systems using either Fat-Fast neurons or standard VG-RAM neurons. For that, we used well known datasets of handwritten digits and traffic signs, and employed well defined metrics for performance evaluation. As our systems have parameters that need to be properly adjusted, we used a parameter tuning procedure for each system before the experimental evaluation.

All experiments were run in a Dell Precision R5500 machine, with 2 Intel Xeon processors of 2.13 GHZ, and 24 GB of DDR3 RAM of 1.33 GHZ, running Ubuntu 12.4LTS. The recognition systems were written in C, and compiled with gcc 4.6.3 with maximum optimization and using OpenMP (the main loops of the code were annotated with proper OpenMP directives). We computed the Hamming distances by doing one xor operation for every 32 bits, followed by a pop count to count the number of 1s – we used the built-in gcc function __builtin_popcountll for this purpose.

3.1 Handwritten Digit Recognition System

The handwritten digit recognition system employs a single layer VG-RAM WNN architecture where each neuron has a set of synapses that are connected to the network's Input Layer (it is identical to the architecture of Figure 1). The synaptic interconnection pattern of each neuron forms a Receptive Field that follows a random two-dimensional Gaussian distribution of synapses equivalent to the one used in the feature-based neural network architecture described in [21], where the center of the Receptive Fields of the neurons are linearly distributed along the Input Layer. The Gaussian synaptic interconnection pattern mimics biological synaptic interconnection patterns observed in many classes of biological neurons [33]; it is randomly created when the network is built and it does not change afterwards.

The WNN synapses can only get a single bit from the network's Input Layer. Thus, in order to allow our WNN to deal with images, in which a pixel may assume a range of different values, we use minchinton cells [34]. Each neuron's synapse forms a minchinton cell with the next (the last one forms a minchinton cell with the first one). The type of the minchinton cell we have used returns 1 (one) if the synapse is connected to an element of the Input Layer whose value is larger than the value of the element to which the next synapse is connected; otherwise, it returns zero.

The images of handwritten digits are transformed before being copied to the network's Input Layer. They are: (i) scaled to fit into the network's input; and (ii) filtered by a Gaussian filter to smooth out artifacts produced by the scaling. During training, a handwritten digit image is transformed (scaled and filtered), the pixels of the transformed image are copied to the network's Input Layer, and all neurons' outputs are set to the value of the label (class id) associated with the image. All neurons are then trained to output this label with this input image. This procedure is repeated for all images of handwritten digits in the training dataset. During testing, the transformed image of a handwritten digit is copied to the network's Input Layer and all neurons' outputs are computed. The number of votes for each label is calculated as the sum of the

neurons outputting that label; the network's output is given by the top ranked label. This procedure is repeated for all images of handwritten digits in the test dataset.

3.2 Traffic Sign Recognition System

The traffic sign recognition system also employs a single layer VG-RAM architecture with the synaptic interconnection pattern of each neuron following a two-dimensional Gaussian distribution. However, the center of the Receptive Field of the neurons is not linearly distributed along the input. Instead, it is centered in the position of the Input Layer given by the inverse log-polar function of the neuron's position in the neural layer (see [24] for details). The VG-RAM synapses collect binary input vectors from the network's Input Layer using minchinton cells [34].

The traffic sign images are transformed before being copied to the network's Input Layer. They are: (i) equalized to improve contrast; (ii) cropped to keep only the region of interest (traffic sign region); (iii) translated to bring the region of interest's center closer to the input image's center; (iv) scaled to fit into the network's Input Layer; (v) filtered by a Gaussian filter to smooth out artifacts produced by transformations; and (vi) filtered by mask filters to yield grayscale images representing each one of the RGB color channels. In contrast to the system proposed in [24], only the grayscale image representing the channel with the highest accuracy (the green channel) was chosen to run most of the experiments presented here.

Training and test procedures were identical to those used in [24], except that, during test, each image is tested only once (see [24] for details).

3.3 Datasets

We compared the performance of the Fat-Fast VG-RAM WNN against that of the standard one using two datasets: (i) the modified National Institute of Standards and Technology (MNIST) handwritten digit dataset [35], and (ii) the German Traffic Sign Recognition Benchmark dataset (GTSRB) [36]. The MNIST handwritten digit dataset is one of the most important benchmark datasets for the machine learning community until today. It includes thousands of segmented grayscale-image samples for each digit class, which constitutes a tough challenge for our new VG-RAM approach improving testing-time whilst maintaining a good compromise with accuracy. The same is true for the GTSRB dataset, which includes hundreds of non-segmented colored-image samples for each traffic sign class. Even though it is quite recent, it is a well-established benchmark dataset in the machine learning community. Although our new VG-RAM approach can be applied to other different machine learning problems, we are mainly interested in image-based classification problems with larger datasets and these two represent very well this type of problem.

3.3.1 MNIST Handwritten Digit Dataset

The MNIST dataset (http://yann.lecun.com/exdb/mnist/) [35] is composed of 70,000 grayscale images of handwritten digits. It is divided in two parts, a 60,000-images training dataset and a 10,000-images test dataset. These images were size normalized to fit into a 20×20 box and centered in a 28×28 image by computing the center of mass of the pixels and translating the image to position this point at the center

of the 28×28 image. Figure 4 shows examples of handwritten digits randomly chosen from the MNIST training dataset.

Figure 4: Examples of handwritten digits of the MNIST dataset [35].

3.3.2 GTSRB Dataset

The GTSRB (http://benchmark.ini.rub.de/) [36, 37] consists of 51,839 colored images of German traffic signs classified into 43 classes. These images contain a border of about 10% around the actual traffic sign (at least 5 pixels) and the traffic sign is not necessarily centered within the image. Image sizes vary between 15×15 and 250×250 pixels. The GTSRB is divided into a training dataset that contains 39,209 images, and a test dataset with 12,630 images. Figure 5 shows examples of the 43 traffic sign classes, which were randomly selected from the GTSRB training dataset.

3.4 Metrics

We compare the performances of the standard VG-RAM against that of Fat-Fast VG-RAM in terms of memory usage, classification precision, and classification test time. We measured the memory usage in bytes with the following equations:

$$M_{vg\text{-}ram} = (p / b + sizeof(t)) \times m \times n \quad (1)$$

$$M_{fat\text{-}fast} = M_{vg\text{-}ram} + h \times (2^{p/h} + m) \times sizeof(m) \times n \quad (2)$$

In Equations (1) and (2), p is the number of synapses, sizeof(.) computes the size in bytes of its argument, m is the number of training samples, n is the number of neurons, and h is the number of hash tables of H. As Equation (1) shows, the memory consumption of the standard VG-RAM, $M_{vg\text{-}ram}$, is equal to the number of bytes necessary to store one input-output pair, L = (I, t), where I requires p / b bytes and t requires sizeof(t) bytes (in our experiments, t is represented by an integer, i.e. sizeof(t) = 4 bytes; and b is the number of synapses represented by a byte), times the number of learned pairs, m, times the number of neurons, n. The memory consumption of the Fat-Fast VG-RAM, Equation (2), is equal to $M_{vg\text{-}ram}$ plus the number of bytes necessary to store the multi-indexed hash table H, which can be inferred by inspecting Figure 2.

We measured the classification precision by counting the number of hits in the test dataset and dividing it by the size of this dataset. The time spent in classification task – i.e. finding the nearest pattern in memory – was estimated by dividing the time it took to classify all samples of test dataset by the number of samples of this dataset. For each classification system implemented, we computed classification time while using standard and Fat-Fast VG-RAM neurons. To compare the performances in terms of time

of these two cases, we computed the speedup obtained with the use of Fat-Fast neurons dividing the classification time with standard by the classification time with Fat-Fast VG-RAM neurons.

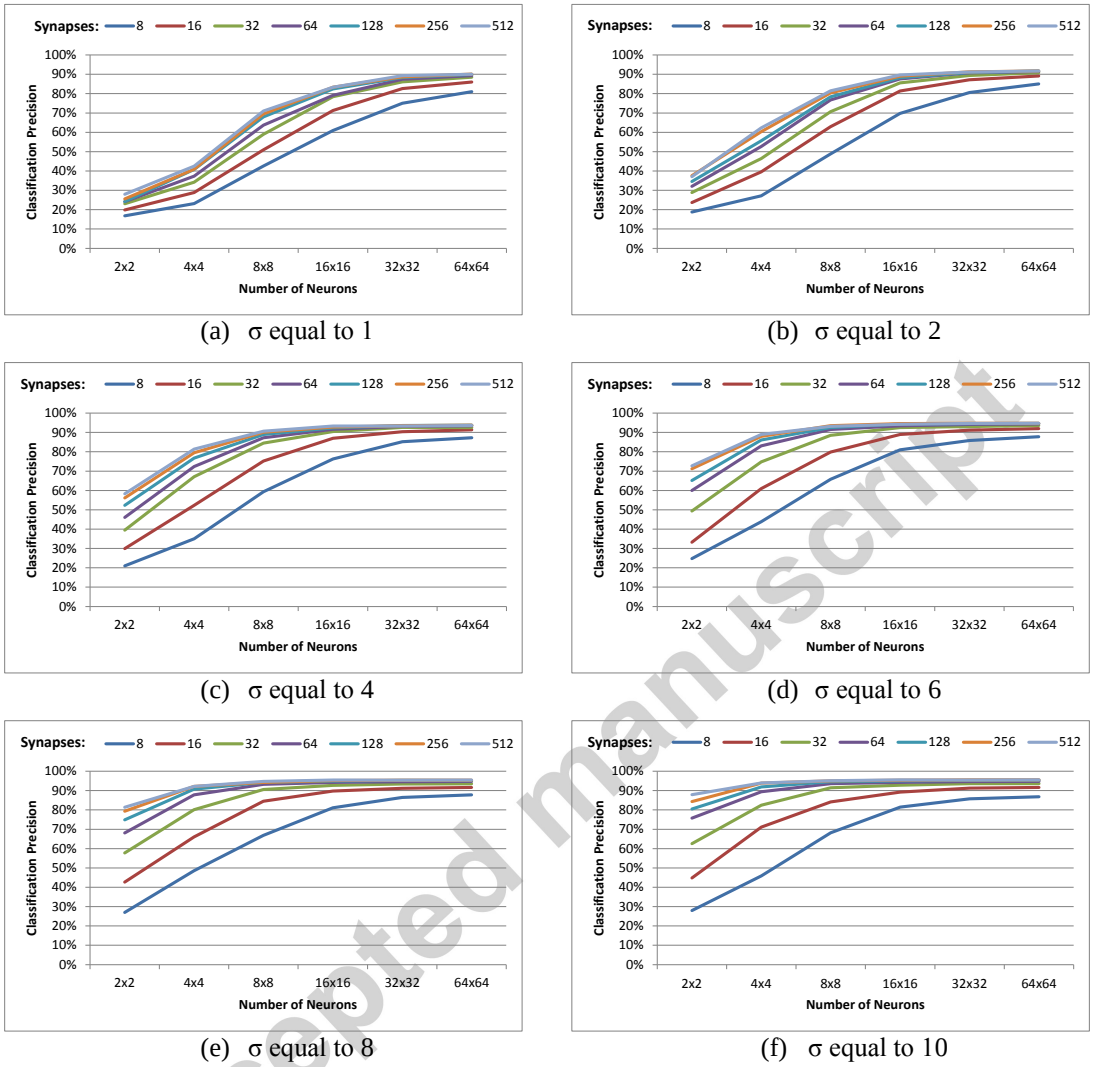
Figure 5: Examples of the 43 traffic sign classes of the GTSRB dataset [36, 37].

3.5 Tuning Parameter Selection

The neural architecture employed in the Handwritten Recognition System has four parameters: (i) the dimensions of the Neural Layer (Figure 1); (ii) the dimensions of the Input Layer; (iii) the number, p, of synapses per neuron; and (iv) the standard deviation, σ, of the Gaussian synaptic interconnection pattern of the neurons [21]. We examined networks with: Neural Layer dimensions equal to 2×2, 4×4, 8×8, 16×16, 32×32, and 64×64 neurons; Input Layer with always twice the size of the Neural Layer; number of synapses per neuron, p, equal to 8, 16, 32, 64, 128, 256 and 512 synapses; and σ equals to 1, 2, 4, 6, 8, and 10.

For tuning the architecture, we randomly selected 15,000 samples from the training dataset, and divided it into 10,000 training samples and 5,000 test samples. As Figure 6 shows, the classification performance of the system grows with the number of

neurons and synapses per neuron, and reaches a plateau at about 32×32 neurons and 128 synapses for all σ values tested. The performance also grows with σ and reaches a plateau at about 10. Considering these results, we selected the number of neurons equal to 32 × 32; the number of synapses per neuron equal to 128; the size of the network input equal to 64 × 64; and the σ equal to 10 pixels.



(a) σ equal to 1 (b) σ equal to 2

(c) σ equal to 4 (d) σ equal to 6

(e) σ equal to 8 (f) σ equal to 10

Figure 6: Handwritten digit recognition system parameter tuning. (a) Classification performance for σ equal to 1. (b) Classification performance for σ equal to 2. (c) σ equal to 4. (d) σ equal to 6. (e) σ equal to 8, (f) σ equal to 10.

We have set the traffic sign recognition system's parameters with the same values presented in [18], i.e.: 51 × 27 neurons, 64 synapses per neuron, and standard deviation σ = 7. For more details regarding the traffic sign recognition system and its parameters tuning please refer to [18].

4. Results

Herein are presented the results obtained according to the experimental methodology described in the previous section.

4.1 Fat-Fast VG-RAM Memory Consumption

The additional memory consumption imposed by the Fat-Fast neuron can be visualized in Table 1 (see equations (1) and (2)). In Table 1, the first column describes the classification system under analysis, the intermediate columns are the parameters of equations (1) and (2) for each system, and the last column is the memory consumption of the system. To take advantage of the ability of current machines to handle 8-bit and 16-bit addresses, in our implementations, the sub-vector V_k size (p/h) was restricted to these two sizes. As Table 1 shows, the use of Fat-Fast neurons increases memory consumption by more than 400% for the handwritten digit recognition system (4,938/1,172 ≈ 421%; and 5,095/1,172 ≈ 435%), and more than 300% for the traffic sign recognition system (6,807/1,854 ≈ 367%; and 5,702/1,854 ≈ 308%). It is interesting to note that the memory consumption of the Fat-Fast implementations with sub-vector V_k size (p/h) equal to 8 bits or 16 bits, either increase memory consumption with the value of p/h (for the Handwritten recognition system), or decrease with the value of p/h (for the Traffic Sign recognition system). This is expected, however, for these values of the ratio p/h, and number of training samples (please examine equation (2)).

Table 1: Memory consumption of Fat-Fast and standard VG-RAM neurons in the Handwritten and the Traffic Sign recognition systems.

Recognition System Type / Neuron Type / Value of p/h	Number of Synapses (p)	Size of int in Bytes	Number of Trained Samples (m)	Number of Neurons (n)	Number of Hash Tables (h)	Memory Size in MB
Handwritten / VG-RAM / –	128	4	60000	1024	–	1172
Traffic Sign / VG-RAM / –	64	4	117627	1377	–	1854
Handwritten / Fat-Fast VG-RAM / 8-bits	128	4	60000	1024	16	4938
Handwritten / Fat-Fast VG-RAM / 16-bits	128	4	60000	1024	8	5095
Traffic Sign / Fat-Fast VG-RAM / 8-bits	64	4	117627	1377	8	6807
Traffic Sign / Fat-Fast VG-RAM / 16-bits	64	4	117627	1377	4	5702

4.2 Fat-Fast VG-RAM Precision and Speedup

In this section, we evaluate the classification precision and speed up of our systems implemented with Fat-Fast neurons, against those of our systems implemented with standard VG-RAM neurons.

4.2.1 Handwritten Digit Recognition System

The bars in the graph of Figure 7 show the precision of standard and Fat-Fast VG-RAM on the classification of the MNIST test dataset for p/h = 16 bits. There is one bar for standard VG-RAM, and several bars for Fat-Fast VG-RAM, each for a different

value of the parameter r (see Section 2.2). The precision value is denoted in the left vertical axis. As the bars in the graph of Figure 7 show, Fat-Fast neurons causes only a small reduction in the classification precision – less than 1% (in about 97%) – for r varying from 30 to 5. With r = 1, the loss in precision is equal to 1.87%.

The curve in the graph of Figure 7 shows the speedup obtained with Fat-Fast neurons; the speedup value is denoted in the right vertical axis. As the curve in Figure 7 shows, for large values of r the speedup is modest. However, for r = 5 the speed is about 40 and for r = 1, higher than 150.

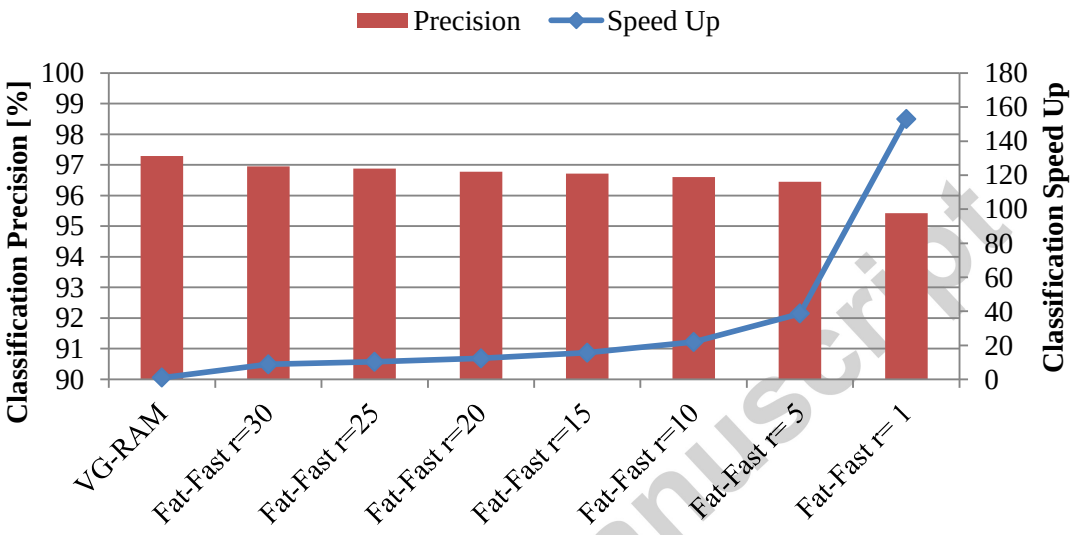


Figure 7: Handwritten Recognition System performance. Bars: Classification precision for implementations with standard VG-RAM, and Fat-Fast VG-RAM with different values of r for p/h = 16 bits. Curve: Speedup of the Fat-Fast implementation for different values of r.

Figure 8 shows our experimental results for the Fat-Fast VG-RAM implementation with sub-vector V_k size (p/h) equal to 8 bits. A comparison of the results presented in Figure 7 and Figure 8 shows that the p/h = 16 bits version is more accurate than the p/h = 8 bits version, especially for r = 1. In addition, the p/h = 16 version is roughly 2 times faster than the p/h = 8 bits version. By comparing the amount of memory consumed by both Fat-Fast architectures (see Table 1), it is possible to see that the p/h = 16 version consumes about the same amount of memory of the p/h = 8 version (5,095 MB versus 4,938 MB). Considering these results, the p/h = 16 version is much preferred. These results highlight the importance of the p/h parameter. It is important to note, however, that increasing this parameter to, say, p/h = 32, would not be viable for current machines, since this would imply hash tables, H_k, of 2^{32} entries each.

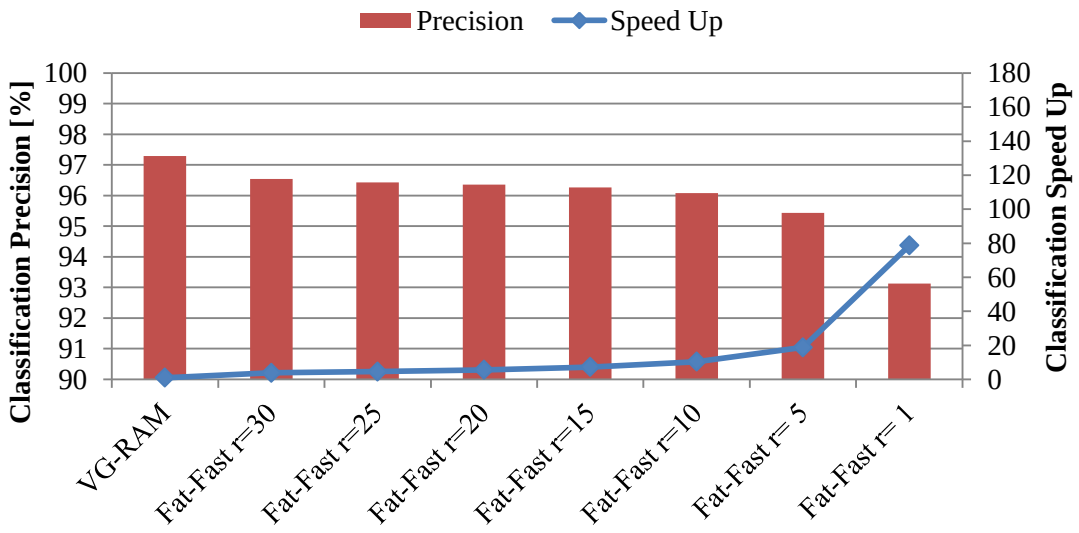


Figure 8: Handwritten Recognition System performance. Bars: Classification precision for implementations with standard VG-RAM, and Fat-Fast VG-RAM with different values of r for p/h = 8 bits. Curve: Speedup of the Fat-Fast implementation for different values of r.

4.2.2 Traffic Sign Recognition System

The bars in the graph of Figure 9 show the precision of standard and Fat-Fast VG-RAM systems on the classification of the GTSRB test dataset for p/h = 16 bits. We show precision bars for all three RGB-color channels (see Section 3.2). The bars in the graph are grouped by neuron type and also by the parameter r for the Fat-Fast system. There is one group of bars for the standard VG-RAM system, and several groups of bars for the Fat-Fast VG-RAM system, each group of the later for a different value of the parameter r (see Section 3). The precision value is denoted in the left vertical axis. As the bars in the graph of Figure 9 show, Fat-Fast neurons causes only a small reduction in the classification precision – about 1% for all three color channels – for r varying from 30 to 5.

The curve in the graph of Figure 9 shows the speedup obtained with Fat-Fast neurons; the speedup value is denoted in the right vertical axis. As the curve in Figure 9 shows, for large values of r, the speedup is about 50 for all channels, and for r = 5, the speedup is about 150. For r = 1, the speedup is higher than 550.

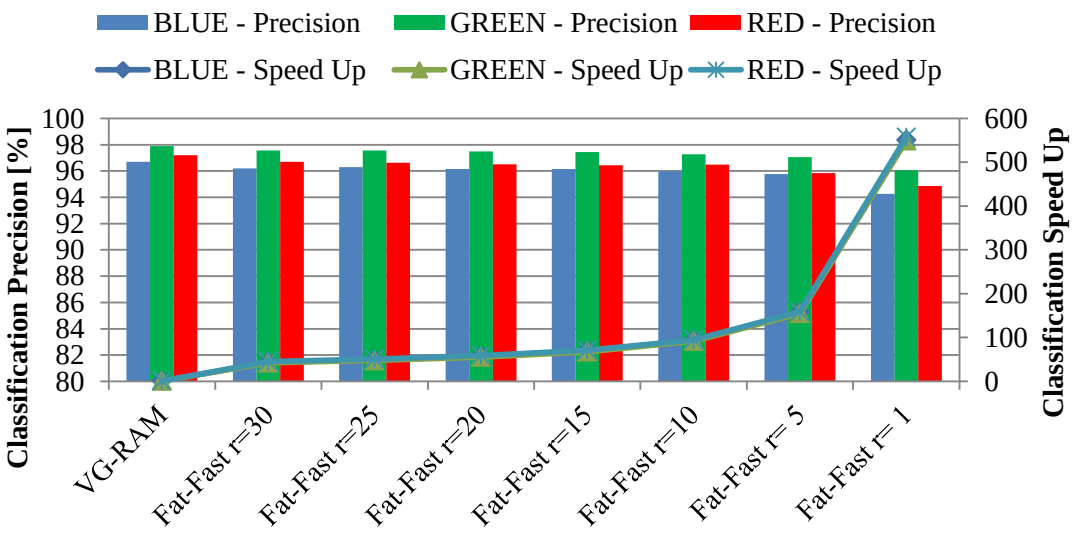


Figure 9: Traffic Sign Recognition System performance. Bars (blue, green, and red channels): Classification precision for implementations with standard VG-RAM, and Fat-Fast VG-RAM with different values of r for p/h = 16 bits. Curve: Speedup of the Fat-Fast implementation for different values of r.

Figure 10 presents the performance of the Fat-Fast VG-RAM WNN with p/h = 8. A comparison of the results in Figure 9 and Figure 10 shows that the p/h = 16 bits version is more accurate than the p/h = 8 bits version, especially for r = 1. In addition, the p/h = 16 version is roughly 2 times faster than the p/h = 8 version.

Overall, the experimental results obtained with Fat-Fast neurons in both classification systems show the same behavior, which suggests that equivalent results should be expected when using Fat-Fast neurons in other applications.

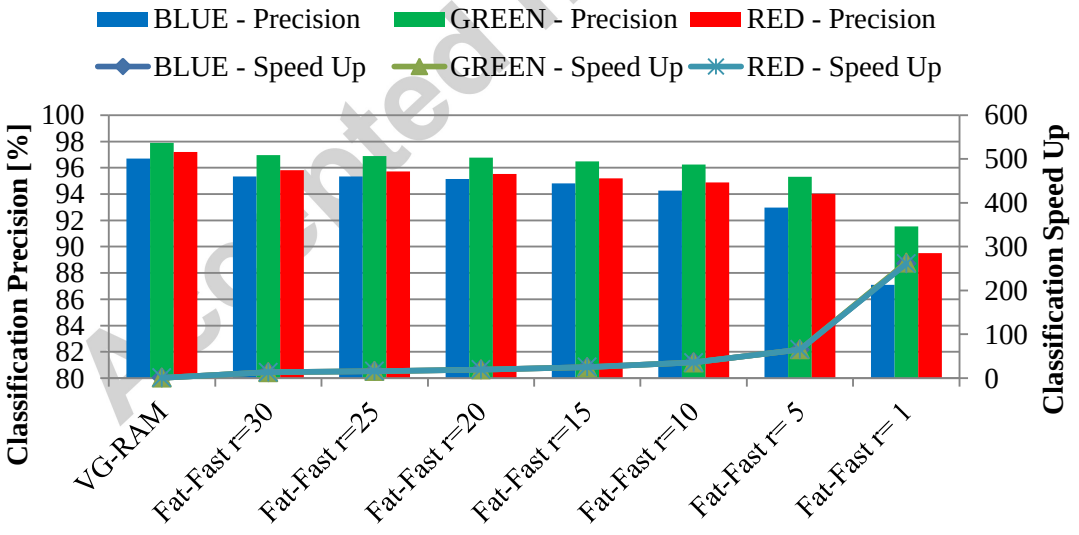


Figure 10: Traffic Sign Recognition System performance. Bars (blue, green, and red channels): Classification precision for implementations with standard VG-RAM, and Fat-Fast VG-RAM with different values of r for p/h = 8 bits. Curve: Speedup of the Fat-Fast implementation for different values of r.

4.3 Performance of Fat-Fast VG-RAM with Memory Reduction

In the next experiments, we evaluated the combined impact of using Fat-Fast VG-RAM neurons and the VG-RAM memory reduction technique presented in Section 2.3 on VG-RAM performance. We measured Fat-Fast precision and speedup for different memory retention levels, for sub-vector V_k size (p/h) equal to 16 and bits 8, and for r equal to 30 and 1. We also examined the impact of memory reduction on standard VG-RAM WNN.

4.3.1 Handwritten Digit Recognition System

The bars of the graph of Figure 11 show the precision of the standard and Fat-Fast VG-RAM implementations of our handwritten digit recognition system for different memory retention levels, rl. We examined five memory retention levels: rl = 0.25%, rl = 0.5%, rl = 1%, rl = 10% and, when no memory reduction has been made, rl = 100%. The Fat-Fast bars are for a system with p/h = 16 bits, and r = 30. As can be seen in Figure 11, the impact on precision of a reduction to 10% of the original memory size (or training set size, m; see Section 2.3) is very small for both, standard and Fat-Fast VG-RAM systems; a reduction to 1% of the original memory has still a small impact of on precision – a decrease of about 1%. Larger reduction levels produce higher impact on precision. Please note that the reduction steps employed have different sizes (rl = 100% ➔ rl = 10%, 10-times reduction step; rl = 10% ➔ rl = 1%, 10-times reduction step; rl = 1% ➔ rl = 0.5%, 2-times step; rl = 0.5% ➔ rl = 0.25%, 2-times step).

The curves of the graph of Figure 11 show the speedups our handwritten digit recognition systems for different memory retention levels, rl. As these curves show, Fat-Fast presents better performance only for rl > 10%. For larger reductions in neuron memory size (rl < 10%), standard VG-RAM presents better speedup and precision than Fat-Fast VG-RAM. In addition, using our memory reduction technique, standard VG-RAM memory consumption reduces linearly with rl (see Equation (1)), while the same does not occur with Fat-Fast VG-RAM, since part of the size of the hash tables, H, does not depend on the size of the training set, m (see Equation (2), term $2^{p/h}$). This is illustrated by Figure 12, where we show the memory footprint of the standard and Fat-Fast systems for the retention levels considered. The memory footprint of Fat-Fast VG-RAM reached a plateau at about rl = 1% and does not decrease after this value.

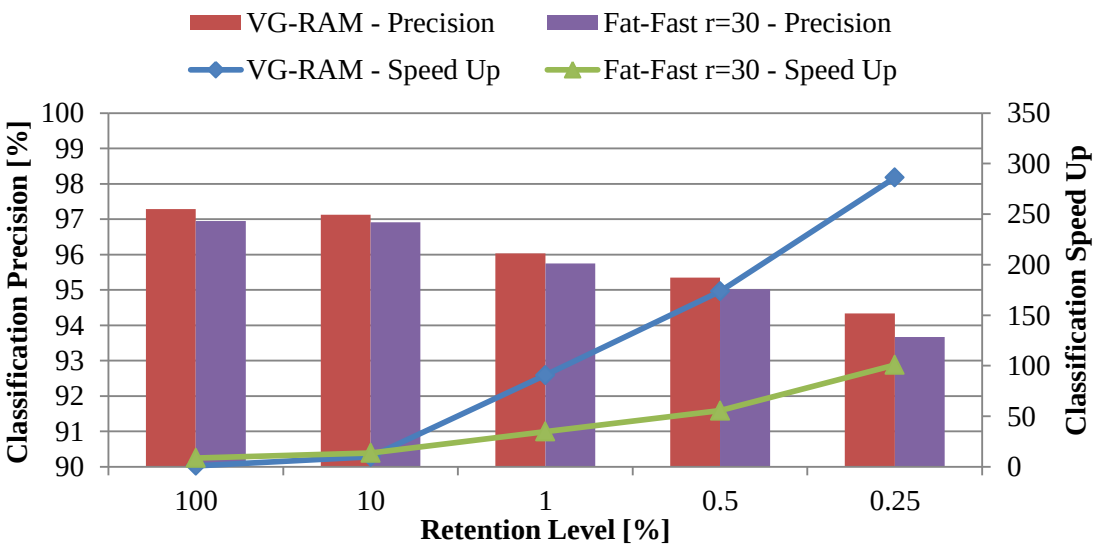


Figure 11: Impact of different memory retention levels on the performance of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 30.

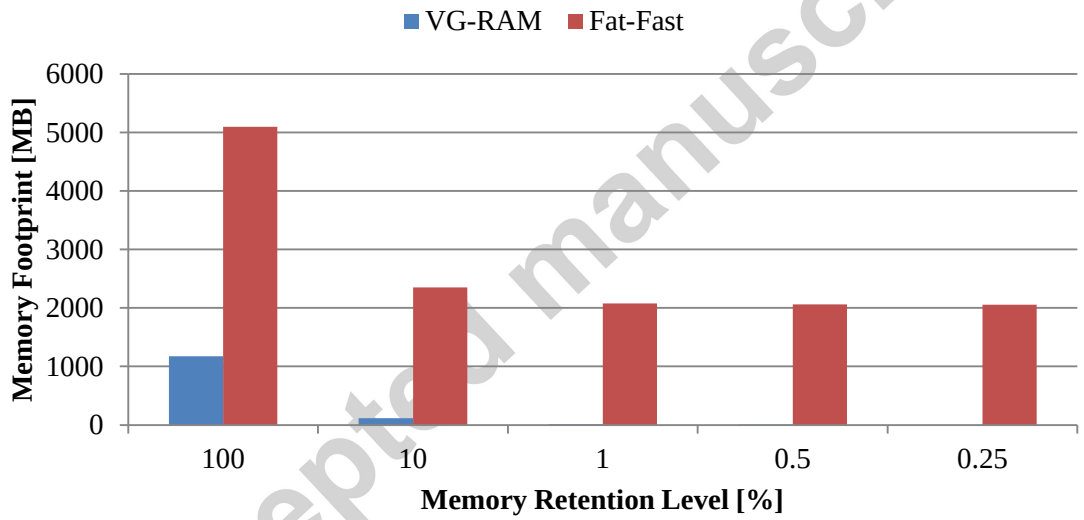


Figure 12: Memory footprint of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 30.

The same experiments were made considering the Fat-Fast system implemented with p/h = 8 bits. The results are in Figure 13. As the bars in this figure show, the classification performance of Fat-Fast VG-RAM with this configuration is closer to that of standard VG-RAM, and even superior for rl = 0.5%. On the other hand, the speedups of Fat-Fast are smaller than those of the configuration with p/h = 16 bits.

Figure 14 shows the memory footprint of the Fat-Fast systems with p/h = 8 bits (the memory footprint of the standard VG-RAM is also shown for comparison, but it is the same of the Figure 12, obviously). The memory footprint of Fat-Fast VG-RAM reaches a plateau at about rl = 0.5% with significantly smaller memory consumption than before (please compare Figure 14 with Figure 12).

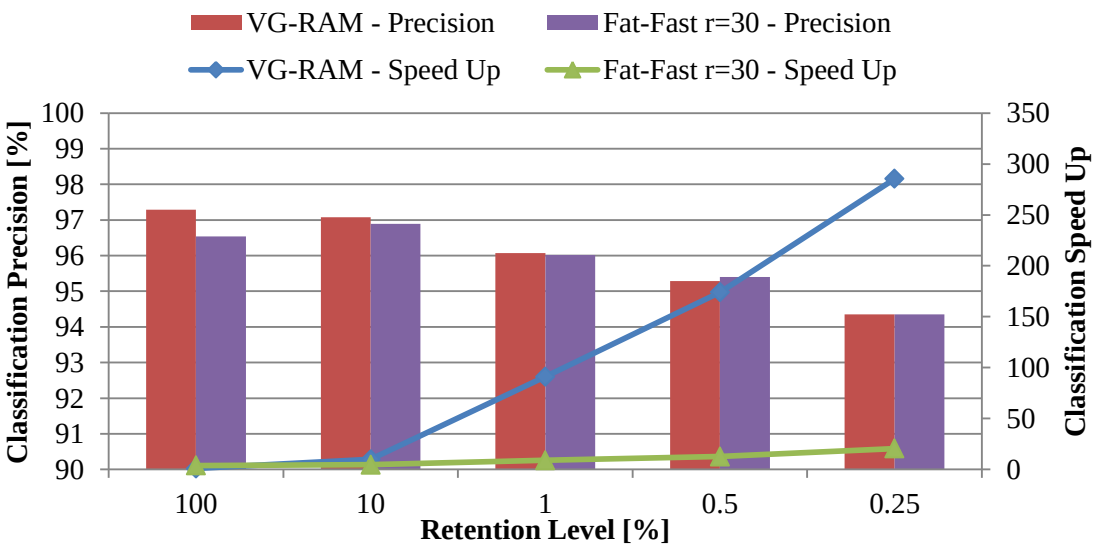


Figure 13: Impact of different memory retention levels on the performance of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 30.

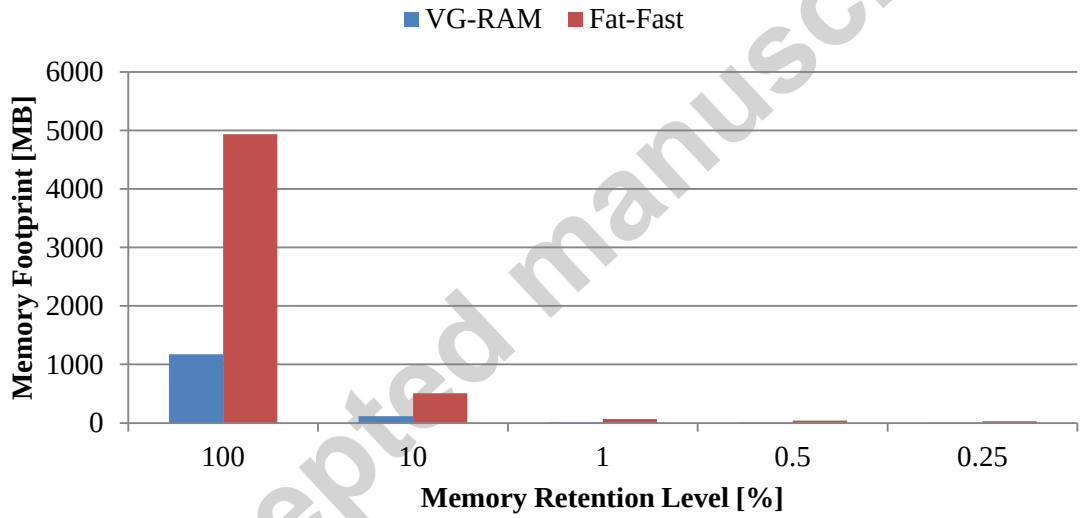


Figure 14: Memory footprint of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 30.

The same experiments were made considering the Fat-Fast system implemented with p/h = 16 and p/h = 8 bits, but with r = 1. Figure 15 and Figure 16 show the results. With r = 1 and p/h = 16 (Figure 15), Fat-Fast is always faster than standard VG-RAM, being able to reach a speedup of 308 for rl = 0.25%. With r = 1 and p/h = 8 (Figure 16), Fat-Fast is faster than standard VG-RAM for rl > 1% only. For both cases, however, Fat-Fast with r = 1 shows significantly worse precision than with r = 30. But it is interesting to note that, for r = 1, Fat-Fast achieves better precision with smaller rl, reaching a plateau at about rl = 1%. This phenomenon can be understood through the following example.

Suppose that an entry v in a hash table H_k of a neuron has a considerable amount of collisions, i.e. a large bucket size, $|B_{k,v}|$ (see Figure 2), and that the current input vector of the neuron has a sub-vector, V_k, addressing this bucket. In this case, there

would be only a single chance (considering r = 1) of choosing the entry in $B_{k,v}$ that points to the memory line of the neuron that has the lowest hamming distance to the input vector. This example illustrates the benefits of our memory reduction technique. It reduces the size the memory, |L|, leaving the most promising training samples in it, and, as a consequence, it also reduces the size of the buckets $B_{k,v}$, leaving in them pointers to promising training samples.

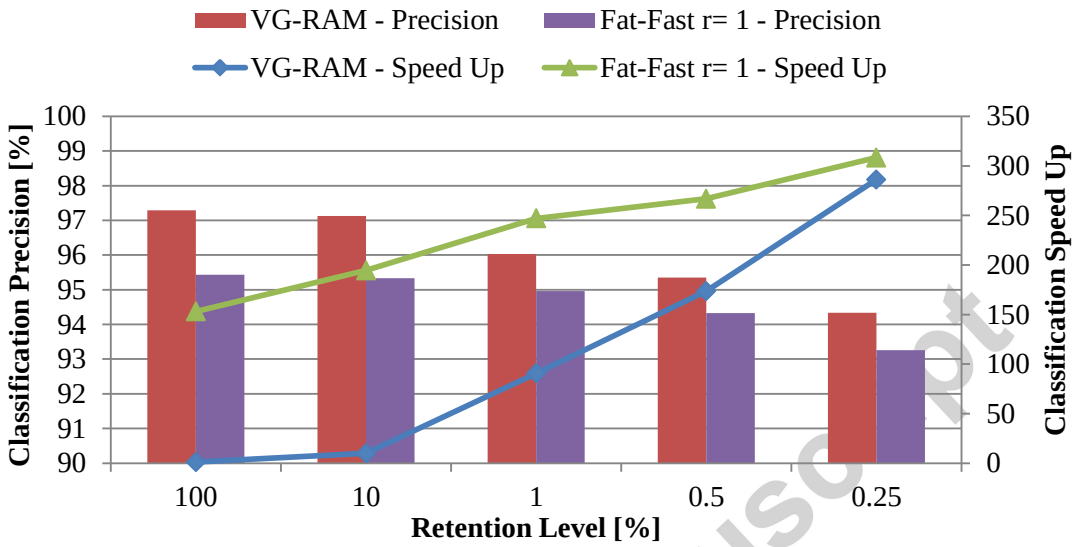


Figure 15: Impact of different memory retention levels on the performance of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 1.

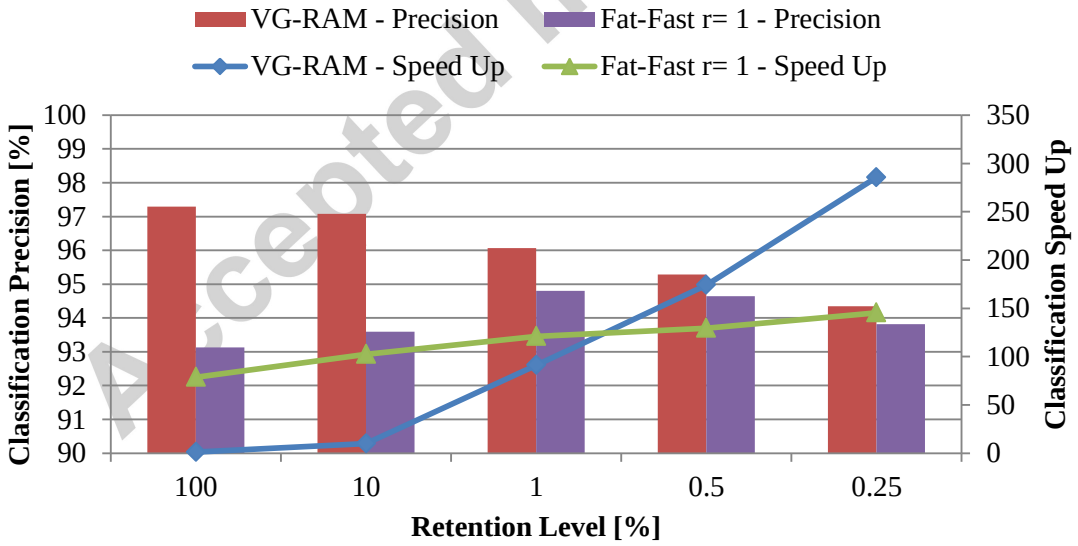


Figure 16: Impact of different memory retention levels on the performance of the Handwritten Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 1.

4.3.2 Traffic Sign Recognition System

In this subsection, we evaluate the combined impact of using Fat-Fast neurons and our memory reduction technique on the performance of the traffic sign recognition system. For clarity, only the green color channel is considered. So, the figures follow the same format as the figures of the previous sub-section; however, please note that the vertical scales are different.

The bars of the graph of Figure 17 show the precision of the standard and Fat-Fast (p/h = 16 bits, r = 30) VG-RAM for different memory retention levels, rl. As can be seen in Figure 17, the impact on precision of a reduction to 10% of the original memory size is small for both, standard and Fat-Fast VG-RAM implementations, but a reduction to 1% of the original memory has a more noticeable impact on precision than before – a decrease of about 2%. Larger reduction levels produce even higher impact on precision.

The speedups achieved with the traffic sign recognition system with the p/h = 16 bits and r = 30 configuration are much higher, as Figure 17 shows. The Fat-Fast implementation is faster for all retention levels and reaches a three order of magnitude speedup over standard VG-RAM without memory reduction – 1,301 times faster for rl = 0.25%. However, this level of retention has a high impact on precision (about 8% loss in precision). Compared with the standard VG-RAM with the same retention level, Fat-Fast is 3.8 times faster and 2.76% less precise.

Figure 18 shows the memory footprint of the standard and Fat-Fast systems for the retention levels considered. Again, the memory footprint of Fat-Fast reaches a plateau at about rl = 1% and does not decrease after this value. However, the memory footprint reduction with smaller retention levels is more noticeable for the traffic sign system (please compare Figure 18 with Figure 12).

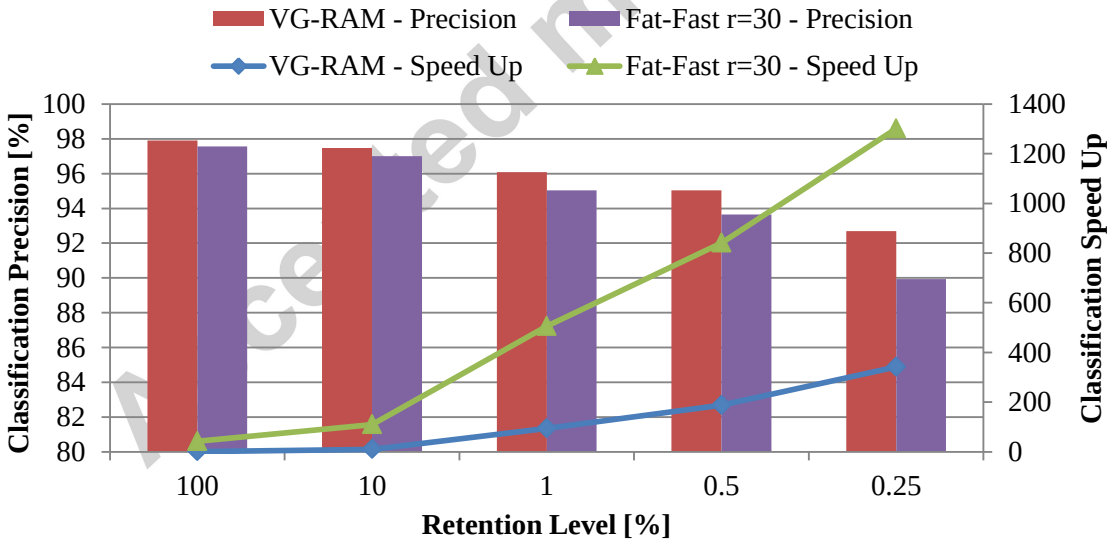


Figure 17: Impact of different memory retention levels on the performance of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 30.

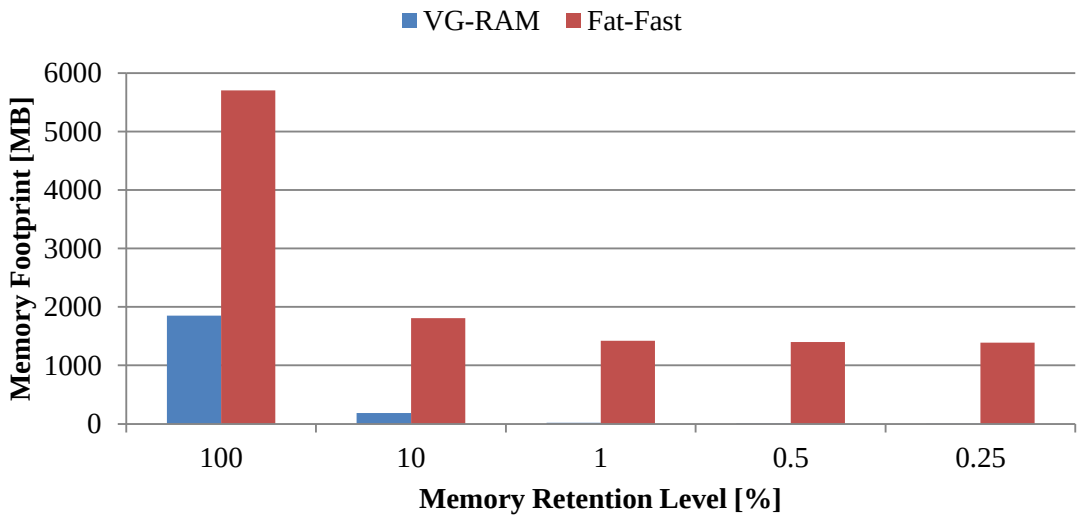


Figure 18: Memory footprint of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 30.

Figure 19 shows the experimental results considering a Fat-Fast system with p/h = 8 bits. As the bars in this figure show, the classification performance of the Fat-Fast VG-RAM with this configuration is closer to that of standard VG-RAM, having about the same performance for rl ≤ 1%. On the other hand, the speedups of Fat-Fast are much smaller than those of the configuration with p/h = 16 bits. Figure 20 shows the memory footprint of the Fat-Fast systems with p/h = 8 bits. The memory footprint of Fat-Fast VG-RAM reaches a plateau at about rl = 1% with significantly smaller memory consumption than before (please compare Figure 20 with Figure 18).

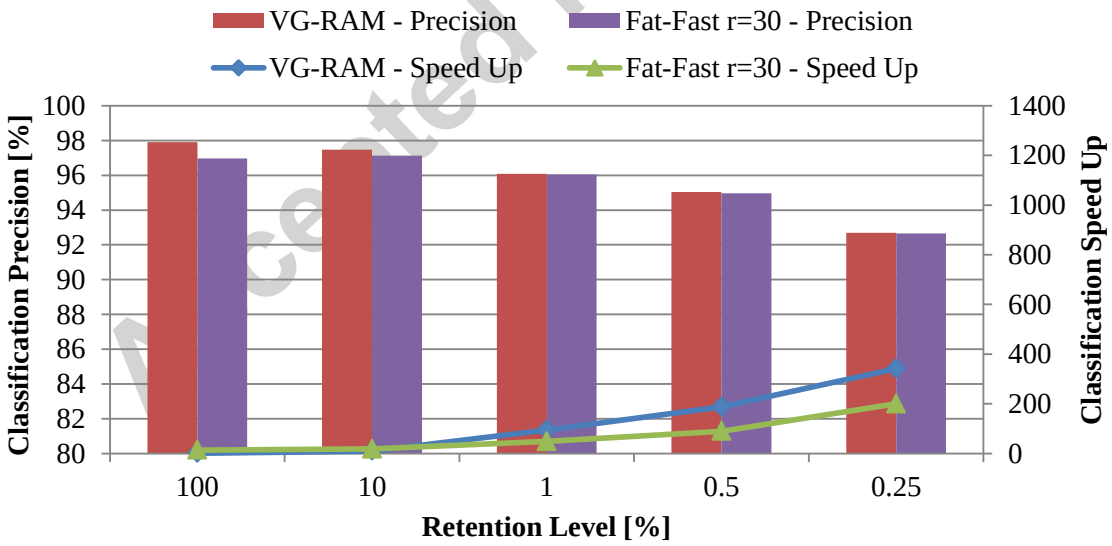


Figure 19: Impact of different memory retention levels on the performance of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 30.

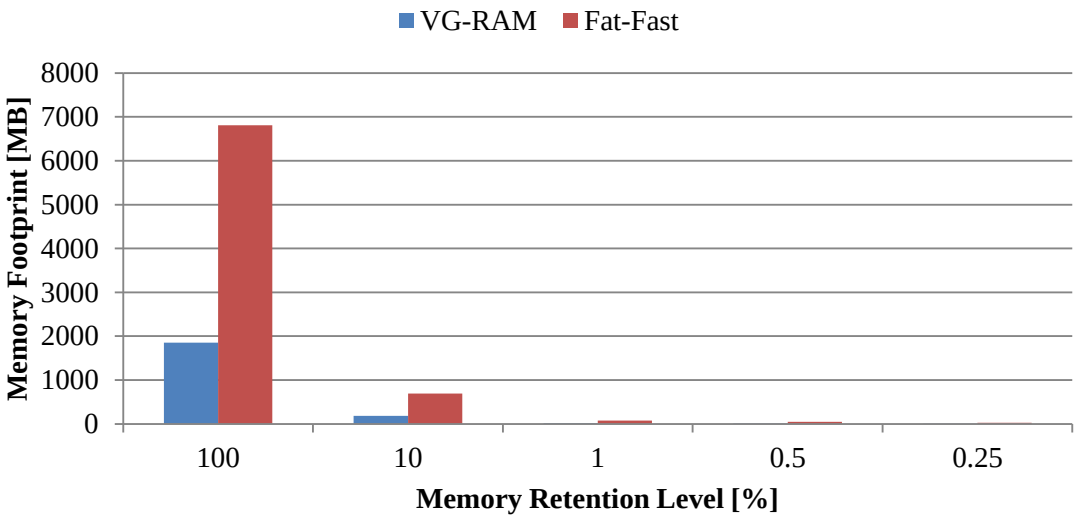


Figure 20: Memory footprint of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 30.

Figure 21 and Figure 22 present the experimental results considering Fat-Fast implementations with p/h = 16 and p/h = 8 bits, and r = 1. Fat-Fast was faster than standard VG-RAM in all experiments with these configurations, reaching a speedup of 1,643 with p/h = 16 for rl = 0.25%. Again, however, Fat-Fast with r = 1 shows significantly worse precision than with r = 30. But it is important to note that, with the p/h = 16 and without memory reduction (rl = 100%), Fat-Fast achieves speedup higher than 500 in this application with less than 2% loss in precision.

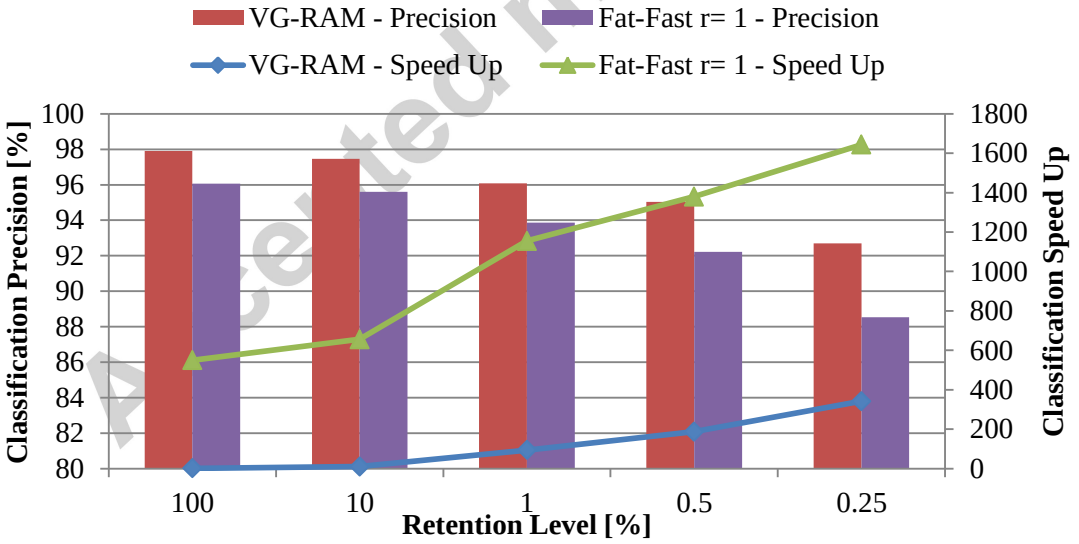


Figure 21: Impact of different memory retention levels on the performance of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 16 bits and r = 1.

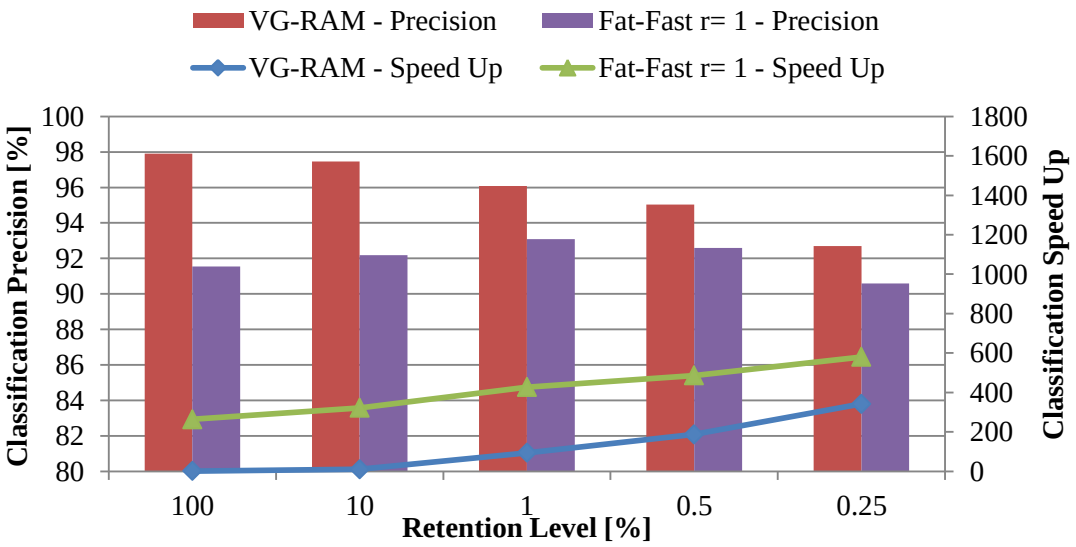


Figure 22: Impact of different memory retention levels on the performance of the Traffic Sign Recognition System implemented with standard VG-RAM, and Fat-Fast VG-RAM with p/h = 8 bits and r = 1.

5. Discussion

The bottleneck of VGRAM based methods is the testing phase. Standard VG-RAM WNN neurons perform a linear search on their training set to find the output value with the nearest input vector. Fat-Fast VG-RAM, in the other hand, exploits the regularity and constancy among bit substrings within its input vectors to perform a faster indexed search. Such an improvement in the technique enables achieving high speedups during the test phase of applications with very large training datasets. Taking the traffic sign application with 117627 training samples as an example, our results showed that the Fat-Fast can speed up the VG-RAM in at least about 50 times with less than 1% loss of accuracy (see Figure 9). In addition, our results also showed that the gain in speed depends on the number of training samples of the application. The handwritten recognition dataset, for example, had a lower speedup (less than 10 times, see Figure 7) because it only has half of the training samples of the traffic sign dataset, and therefore speedup is affected by the overhead of processing the indexation through the hashes. Our experiments also investigated the effect of the parameter r. Parameter r is able to control the gain in speed, because it imposes an upper bound limit on the number of potential comparisons to be done by the Fat-Fast VG-RAM neuron, in case of large number of collisions occurs in the multi-indexed hash table. Our traffic sign application results also showed that speedups higher than 550 times can be achieved by adjusting the parameter r = 1 of the proposed method, but compromising the accuracy in up to 2% (see Figure 9).

As mentioned before, a drawback of the Fat-Fast method is the increase of the used memory because of the multi-indexed hash tables. Applications running on desktop, with good memory capacity, impose no additional problem. However sometimes, it is desirable to keep the use of the memory to a certain level following the constraints of a specific machine. To cope with such cases and to reduce even more the testing time of the applications, we also propose to use a memory reduction technique [26] together with the fat-fast approach. The effect of the Fat-Fast method in the memory size is evident in all our experiments, and is highlighted in Figure 12, Figure 14, Figure 18 and Figure 20. As it can be seen, there is clearly a trade-off between

memory consumption, speedup and machine learning performance. The figures show that even with our memory reduction technique, the Fat-Fast VG-RAM WNN may require much more memory than standard VG-RAM WNN. The benefits of the memory reduction are supported by our results, for the traffic sign application for example (see Figure 17). They showed that the memory from the original VG-RAM can be reduced to 10% of the original size and used together with the Fat-Fast to achieve speedups of 100 times with a decrease in accuracy of less than 1% (see Figure 17). Although the gain in speed is now only from 100 times, it is important to realize that the original memory was decreased to 10%. Another important fact is that the Fat-Fast approach has more effect in memories with many samples, and a reduction of the memory also reduces the effect of the Fat-Fast (see Figure 11 for the Handwritten dataset where normal VGRAM is faster than the Fat-Fast). However, the gain in speed can still be seen for large training sets (e.g. retention level of 10%). Finally, other results also showed that higher gains in speed can be achieved on the cost of the accuracy, when using a combination of Fat-Fast and memory compression (e.g. 0.25% of what the original size achieves is 1301 times faster and 8% less precise, see Figure 17 for an example). In summary, the following rules should be used to decide on which technique combination to use. If you have an application with a large training set, but have no restriction on the testing time, use standard VGRAM. Otherwise, if you have restriction on the testing time, but have no restriction on the available memory, use Fat-Fast VGRAM. In addition, adjust the parameter r to get the best accuracy according to your available memory (r controls the trade-off between Fat-Fast memory use and accuracy). Otherwise, if you have restriction on the testing time and have restriction on the available memory, use the combination of Fat-Fast VGRAM and memory compression. In addition, adjust the parameter rl to get the best accuracy according to your available memory (rl controls the trade-off between memory compression and accuracy).

6. Conclusion

In this paper we presented the Fat-Fast VG-RAM WNN. Fat-Fast VG-RAMs employ multi-index chained hashing for fast search in the neuron's memory. The chained hashing technique increases the memory consumption of VG-RAM substantially; however, it strongly reduces test time, while keeping most of the machine learning performance of VG-RAM. To address the larger memory consumption of Fat-Fast VG-RAMs, we have employed data clustering techniques for reducing the overall amount of memory space required to store training data. Using Fat-Fast neurons with memory reduction via clustering techniques, we were able to reduce the standard VG-RAM runtime and memory footprint, while maintaining a high and acceptable machine learning performance.

To validate the Fat-Fast VG-RAM with memory reduction, we performed several experiments with two well-established and large datasets: the MNIST handwritten digit dataset and the GTSRB dataset. Our experimental results showed that our new optimized VG-RAM approach was able to run up to three orders of magnitude faster and consume two orders of magnitude less memory than standard VG-RAM, while experiencing only a small reduction in classification performance. Thus achieving high-performance results in terms of accuracy, runtime and memory consumption.

In future works, we will examine the Fat-Fast VG-RAM performance in other relevant applications, better policies for memory access using the operating system facilities such as posix_madvise, and the implementation of Fat-Fast VG-RAM in hardware (in Field Programmable Gate Arrays – FPGA, for example).

7. Acknowledgements

We would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, Brazil (grant 99999.002147/2014-09), Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Brazil (grants 552630/2011-0, 308096/2010-0, and 314485/2009-0) and Fundação de Amparo à Pesquisa do Espírito Santo – FAPES, Brazil (grant 48511579/2009), for their support to this research work.

8. References

[1] T. Ludermir, A. Carvalho, A. Braga, and M. Souto, "Weightless neural models: A review of current and past works," Neural Computing Surveys, vol. 2, pp. 41–61, 1999.

[2] R. Rohwer and M. Morciniec, "The Theoretical and Experimental Status of the n-tuple Classifier," Neural Networks, vol. 11, no. 1, pp. 1–14, Jan. 1998.

[3] M. Morciniec and and R. Rohwer, "The n-tuple Classifier: Too Good to Ignore," 1995.

[4] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," Neurocomputing, vol. 74, no. 1–3, pp. 239–255, Dec. 2010.

[5] W. W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine," in Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference, New York, NY, USA, 1959, pp. 225–232.

[6] I. Aleksander, W. V. Thomas, and P. A. Bowden, "WISARD: a radical step forward in image recognition," Sensor Review, vol. 4, no. 3, pp. 120–124, Dec. 1984.

[7] I. Aleksander, "From WISARD to MAGNUS: A Family of Weightless Virtual Neural Machines," in Progress in Neural Processing, vol. 9, WORLD SCIENTIFIC, 1998, pp. 18–30.

[8] J. G. D. O. Cardoso, Massimo De Gregorio, Felipe França, Maurizio Giordano, and Priscila Lima, "WIPS: the WiSARD Indoor Positioning System," in Proceedings of ESANN 2013, Bruges, Belgium, 2013, pp. 521–526.

[9] Q. Yao, D. Beetner, D. C. Wunsch, and B. Osterloh, "A RAM-based neural network for collision avoidance in a mobile robot," in Proceedings of the International Joint Conference on Neural Networks, 2003, 2003, vol. 4, pp. 3157–3160 vol.4.

[10] P. Coraggio and M. Gregorio, "WiSARD and NSP for Robot Global Localization," in Proceedings of the 2nd International Work-conference on Nature Inspired Problem-Solving Methods in Knowledge Engineering: Interplay Between Natural and Artificial Computation, Part II, Berlin, Heidelberg, 2007, pp. 449–458.

[11] S. Nurmaini, S. Z. M. Hashim, and D. N. A. Jawawi, "Modular Weightless Neural Network Architecture for Intelligent Navigation," Int. J. Advance. Soft Comput. Appl., vol. 1, no. 1, Jul. 2009.

[12] C. R. Souza, F. F. Nobre, P. V. M. Lima, R. M. Silva, R. M. Brindeiro, and F. M. G. França, "Recognition of HIV-1 subtypes and antiretroviral drug resistance using weightless neural networks," in Proceedings of ESANN 2012, Bruges, Belgium, 2012, pp. 429–434.

[13] H. L. França, J. C. P. da Silva, M. De Gregorio, O. Lengerke, M. S. Dutra, and F. M. G. França, "Movement persuit control of an offshore automated platform via a RAM-based neural network," in 2010 11th International Conference on Control Automation Robotics Vision (ICARCV), 2010, pp. 2437–2441.

[14] D. O. Cardoso, P. M. V. Lima, M. D. Gregorio, J. Gama, and M. G. França, "Clustering data streams with weightless neural networks," in Proceedings of ESANN 2011, Bruges, Belgium, 2011, pp. 201–206.

[15] C. B. do Prado, F. M. G. Franca, E. Costa, and L. Vasconcelos, "A New Intelligent Systems Approach to 3D Animation in Television," in Proceedings of the 6th ACM International Conference on Image and Video Retrieval, New York, NY, USA, 2007, pp. 117–119.

[16] I. Aleksander, M. De Gregorio, F.M.G. França, P.M.V. Lima, and H. Morton, "A brief introduction to weightless neural systems," in Proceedings of ESANN 2011, Bruges, Belgium, 2011, pp. 299–305.

[17] I. Aleksander and H. Morton, An Introduction to Neural Computing. Internat. Thomson Computer Press, 1995.

[18] M. Berger, A. Forechi, A. F. De Souza, J. De Oliveira Neto, L. Veronese, V. Neves, E. de Aguiar, and C. Badue, "Traffic Sign Recognition with WiSARD and VG-RAM Weightless Neural Networks," Journal of Network and Innovative Computing, vol. 1, no. 1, pp. 87–98, 2013.

[19] A. F. De Souza, C. Badue, B. Z. Melotti, F. T. Pedroni, and F. L. L. Almeida, "Improving VG-RAM WNN Multi-label Text Categorization via Label Correlation," in Eighth International Conference on Intelligent Systems Design and Applications, 2008. ISDA '08, 2008, vol. 1, pp. 437–442.

[20] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue, "Automated Multi-label Text Categorization with VG-RAM Weightless Neural Networks," Neurocomput., vol. 72, no. 10–12, pp. 2209–2217, Jun. 2009.

[21] A. F. D. Souza, C. Badue, F. Pedroni, E. Oliveira, S. S. Dias, H. Oliveira, and S. F. de Souza, "Face Recognition with VG-RAM Weightless Neural Networks," in Artificial Neural Networks - ICANN 2008, V. Kůrková, R. Neruda, and J. Koutník, Eds. Springer Berlin Heidelberg, 2008, pp. 951–960.

[22] A. F. De Souza, C. Badue, F. Pedroni, S. Schwanz, H. Oliveira, and S. F. de Souz, "VG-RAM Weightless Neural Networks for Face Recognition," in Face Recognition, M. Oravec, Ed. InTech, 2010.

[23] J. L. Moraes, A. F. D. Souza, and C. Badue, "Facial access control based on VG-RAM weightless neural networks," in Proceedings of the International Conference on Artificial Intelligence (ICAI'2011), 2011, pp. 444–450.

[24] M. Berger, A. Forechi, A. F. D. Souza, J. de Oliveira Neto, L. Veronese, and C. Badue, "Traffic sign recognition with VG-RAM Weightless Neural Networks," in 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012, pp. 315–319.

[25] A. F. De Souza, C. Fontana, F. Mutz, T. Alves de Oliveira, M. Berger, A. Forechi, J. de Oliveira Neto, E. de Aguiar, and C. Badue, "Traffic sign detection with VG-RAM weightless neural networks," in The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–9.

[26] E. de Aguiar, A. Forechi, L. Veronese, M. Berger, A. F. D. Souza, C. Badue, and T. Oliveira-Santos, "Compressing VG-RAM WNN Memory for Lightweight Mobile Applications," in The 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, In Press.

[27] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast Search in Hamming Space with Multi-index Hashing," in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Washington, DC, USA, 2012, pp. 3108–3115.

[28] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast Exact Search in Hamming Space With Multi-Index Hashing," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 99, no. 1, p. 1, Dec. 2013.

[29] R. Xu and I. Wunsch, D., "Survey of clustering algorithms," IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645–678, May 2005.

[30] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, Sep. 1999.

[31] L. Rokach, "A survey of Clustering Algorithms," in Data Mining and Knowledge Discovery Handbook, O. Maimon and L. Rokach, Eds. Springer US, 2010, pp. 269–298.

[32] J. MacQueen, "Some methods for classification and analysis of multivariate observations," presented at the Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 1967.

[33] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. J. Hudspeth, Principles of neural science. New York: McGraw-Hill Medical Publishing Division, 2011.

[34] R. J. Mitchell, J. M. Bishop, S. K. Box, and J. F. Hawker, "Comparison of some methods for processing Grey Level data in weightless networks," in RAM-based neural networks, J. Austin, Ed. Singapore: World Scientific Publishing Co Pte Ltd, 1998, pp. 66–71.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[36] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in The 2011 International Joint Conference on Neural Networks (IJCNN), 2011, pp. 1453–1460.

[37] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition," Neural Netw, vol. 32, pp. 323–332, Aug. 2012.