2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Compressing VG-RAM WNN Memory for Lightweight Applications

Edilson de Aguiar, Avelino Forechi, Lucas Veronese, Mariella Berger,
Alberto F. De Souza, Claudine Badue, and Thiago Oliveira-Santos

***Abstract*—The Virtual Generalizing Random Access Memory Weightless Neural Network (VG-RAM WNN) is an effective machine learning technique that offers simple implementation and fast training. One disadvantage of VG-RAM WNN, however, is the test time for applications with many training samples, i.e. large multi-class classification applications. In such cases, the test time tends to be high, since it increases with the size of the memory of each neuron. In this paper, we present a new methodology for handling such applications using VG-RAM WNN. By employing data clustering techniques to reduce the overall size of the neurons' memory, we were able to reduce the network's memory footprint and the system's runtime, while maintaining a high and acceptable classification performance. We evaluated the performance of our VG-RAM WNN system with compressed memory on the problem of traffic sign recognition. Our experimental results showed that, after compression, the system was able to run at very fast response times in standard computers. Also, we were able to load and run the system at interactive rates in small low-power systems, experiencing only a small reduction in classification performance.**

***Keywords*—Virtual Generalizing Random Access Memory Weightless Neural Networks, Clustering Techniques, Traffic Sign Recognition.**

I. Introduction

Weightless Neural Networks (WNN) do not store knowledge in their connections, but in Random Access Memories (RAM) inside the network's neurons. The synapses of each neuron collect a vector of bits from the network's input (neuron's input vector) and use it as a RAM address to access the respective associated label value (neuron's output value) stored at this RAM address. Training can be made in one shot and basically consists of storing the desired neuron's output value in the address associated with the neuron's input vector [1]. In spite of their remarkable simplicity, WNN are very effective as pattern recognition tools, offering easy implementation in addition to fast training [2]. However, neuron memory size becomes prohibitive if the neuron's input vector is too large, since it is dictated by this vector size (neuron memory size is equal to 2^p, where p is the neuron's input vector size).

The proponents of Wilkes, Stonham and Aleksander Recognition Device (WiSARD) tackled this problem by splitting the p-sized neuron's input vector in q segments, each one used as address of a specific RAM memory module of size $2^{p/q}$ [3]. During training, a p-sized training input pattern is split in q parts and each p/q-sized input sub-pattern is used as address of the corresponding RAM module; the addressed position in each RAM module receives (stores) the label value (RAM module's output value) associated with the input pattern. During test, the p-sized test input pattern is also split in q parts and each p/q-sized input sub-pattern is used as address of the corresponding RAM module; the neuron's output value is the most frequent label value observed among the outputs of the q RAM modules. Thanks to this organization, the total amount of memory required per neuron is reduced from 2^p to $q \times 2^{p/q}$.

The proponents of the Virtual Generalizing Random Access Memory (VG-RAM) Weightless Neural Networks (WNN) took a step further by only requiring memory capacity to store the data related to the training set [4]. Instead of storing an output label value that can be referenced by a binary input pattern, this type of WNN neuron stores pairs of corresponding input-output patterns, i.e. vector of bits as input and label value as output. The memory footprint is optimized with such an approach whereas the performance of training is maintained, facilitating the prototyping of applications for classification. One disadvantage introduced by this method is the test speed of unknown samples, which depends on the size of the neurons' memory (that is equal to the number of trained input-output pairs) – during test, the search for the closest pattern is performed sequentially and requires scanning the whole memory of each neuron.

It has been shown that this type of network has high classification performance for a variety of multi-class classification applications, such as text categorization [5, 6], face recognition [7, 8, 9], and traffic sign detection and recognition [10, 11]. However, although classification speed might be ignored in applications with small training datasets, it becomes a problem in applications with large training datasets. Therefore, the use of this network in systems that demand fast response, or in small low-power systems, is restricted by the number of training samples. Since this number can be quite large for some of the mentioned applications, the sequential search for the closest pattern in the memory of each neuron hinders performance during test

Edilson de Aguiar is with the Department of Computer Science and Electronics, UFES, São Mateus, ES, Brazil.

Avelino Forechi, Lucas Veronese, Mariella Berger, Alberto Ferreira De Souza (e-mail: alberto@lcad.inf.ufes.br), Claudine Badue, and Thiago Oliveira-Santos are with the Department of Informatics, UFES, Vitoria, ES, Brazil.

This work was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq-Brasil (grants 552630/2011-0, 308096/2010-0, and 314485/2009-0) and Fundação de Amparo à Pesquisa do Espírito Santo-FAPES-Brasil (grant 48511579/2009).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

of new samples. Moreover, small low-power systems have further restrictions on available memory, which makes it difficult or even impossible to load networks trained with large training datasets on such devices.

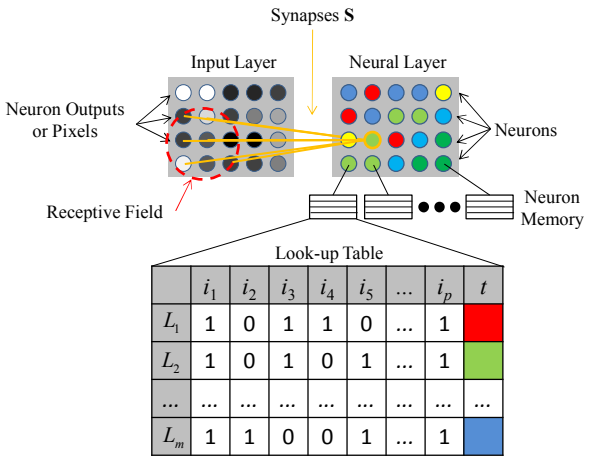
To address these problems and propose a solution for applications based on VG-RAM WNN (VG-RAM for short) with large training datasets, this work investigates the use of data compression techniques to reduce the overall size of the neurons' memory, while keeping a high and acceptable classification performance. Under the hypothesis that not all data stored in the memory of a neuron are relevant for good classification performance of the network, we apply data clustering techniques to eliminate redundant or irrelevant examples from the memory (i.e. to eliminate lines of the look-up table illustrated in Fig. 1). Using a complex multi-class classification application, i.e. traffic sign recognition, we compare the classification performance of VG-RAM with full memory with that of VG-RAM with the memory compressed at different levels. Additionally, as baseline, we compare the classification performance of standard VG-RAM against that of VG-RAM with random deletion of neuron memory samples. Our results show that it is possible to run such heavy multi-class classification applications implemented with VG-RAM at very fast response times on standard computers, or at good interactive levels on small low-power systems.

This paper is organized as follows. After this introduction, in Section II, we briefly present our VG-RAM architecture for multi-class classification, i.e. traffic sign recognition. Subsequently, in Section III, we describe the technique used to reduce the overall size of the neurons' memory, while keeping a good classification performance. In Section IV, we present our experimental methodology and analyze our experimental results. Finally, our conclusions and directions for future work follow in Section V.

II. VG-RAM WNN

The architecture of VG-RAM WNNs comprises neural layers with many neurons connected to input layers (e.g. images or other neural layers) through a set $\mathbf{S} = \{s_1, \ldots, s_p\}$ 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 this vector 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 $\mathbf{L} = \{L_1, \ldots, L_j, \ldots, L_m\}$ 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 Fig. 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	

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

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 test 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 (Fig. 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 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.

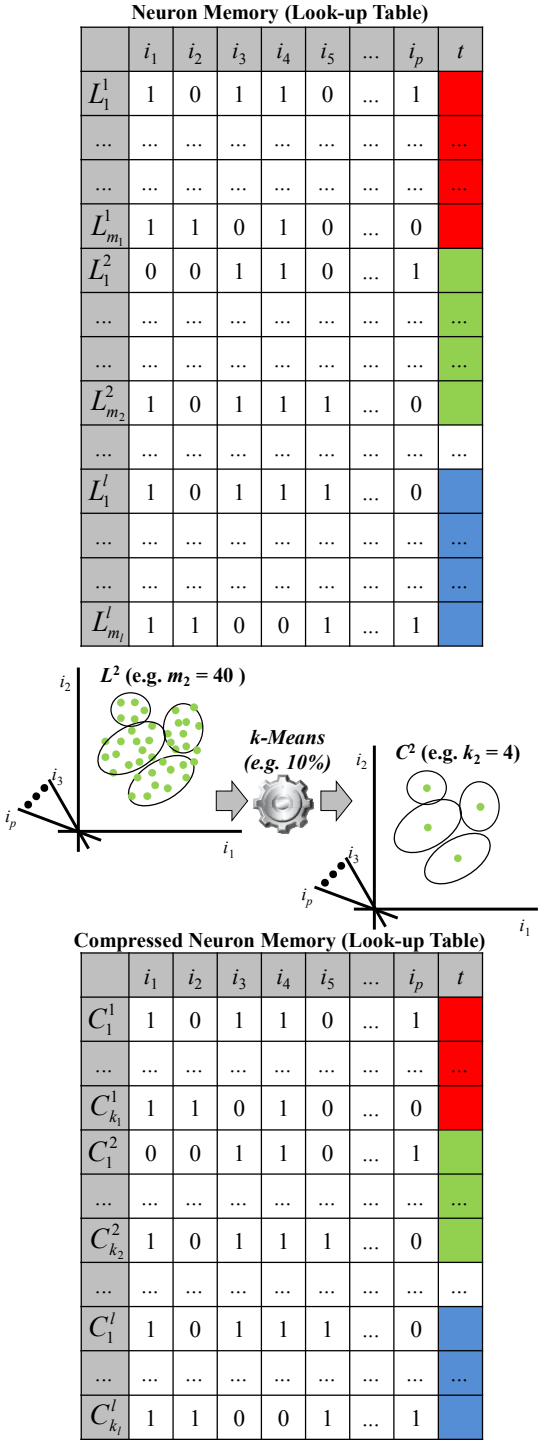


Fig. 2. VG-RAM WNN memory compression framework. The original memory of each neuron, containing m lines, is grouped according to its output type, t. Each group is clustered separately using k-Means and the resulting centroids become the new entries for the compressed memory.

The Hamming distance between two binary patterns can be efficiently computed at machine code level in current machines 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.

III. VG-RAM WNN Memory Compression

Despite the high classification performance of VG-RAM WNN for a variety of multi-class classification applications, the test performance depends on the size of the memory of each neuron. Considering an application requiring a VG-RAM with n neurons, each containing a memory with m input-output pairs, its runtime performance will be proportional to $n \times m$ ($O(nm)$). Unfortunately, many problems require a large training dataset to be solved satisfactorily (e.g. traffic sign recognition [10]). For applications with a large training set, the value of m is large and may limit the use of VG-RAM, since testing a new input sample requires a sequential search for the closest pattern in the m-sized memory of the n neurons.

We aim at reducing the overall size m of the Look-up Tables, thus improving runtime performance and reducing memory footprint, while keeping an acceptable classification accuracy. Assuming that the memory of each neuron will have a large amount of redundant or irrelevant information, we should be able to accomplish relevant compression by carefully eliminating input-output pairs of the neurons' Look-up Tables. This can be achieved by using a clustering algorithm to find the most relevant input-output pairs in the memory of each neuron as described next.

Considering the large number of relevant clustering techniques ([12, 13, 14]) and the fact that the clustering procedure needs to be done many times for each neuron (see below), we decided to use k-Means [15] due to its simplicity and efficiency. The k-Means algorithm partitions a set $\mathbf{I}$ of $|\mathbf{I}|$ vectors into k clusters, $k < |\mathbf{I}|$, where the parameter k is set a priori. This iterative partitioning 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.

Fig. 2 illustrates our VG-RAM memory compression framework that works as follows. Initially, the original memory of each neuron, containing m lines, is sorted according to its output label t (i.e. traffic sign of same type). The result is a set of lines ordered according to each output type, e.g. the set of lines $\mathbf{L}^1 = \{L_1^1, \ldots, L_{m_1}^1\}$ for outputs of type equal to label 1, $\mathbf{L}^2 = \{L_1^2, \ldots, L_{m_2}^2\}$ for outputs of type equal to label 2, until $\mathbf{L}^l = \{L_1^l, \ldots, L_{m_l}^l\}$ for outputs of type equal to label l. Subsequently, we apply k-Means separately to each set of lines $\mathbf{L}^1$, $\mathbf{L}^2$, ..., $\mathbf{L}^l$, 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 $\mathbf{C}^1 = \{C_1^1, \ldots, C_{k_1}^1\}$ of centroids from the lines of set $\mathbf{L}^1$, where $k_1 < m_1$ and $m_1 = |\mathbf{L}^1|$; the set $\mathbf{C}^2 = \{C_1^2, \ldots, C_{k2}^2\}$ of centroids from the lines of set $\mathbf{L}^2$, where $k_2 < m_2$; and so on until the set $\mathbf{C}^l = \{C_1^l, \ldots, C_{k_l}^l\}$, where $k_l < m_l$. The centroid sets $\mathbf{C}^1$, $\mathbf{C}^2$, ..., $\mathbf{C}^l$ become the new memory entries for the respective neuron (see Fig. 2). This process is repeated for all neurons, i.e. the clustering process is performed $n \times l$ times. As a result, the

number of entries in the VG-RAM 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 used different values of k for each label type, i.e. $k_j = m_j \times cl$, where $0 < cl < 1$ is the VG-RAM memory compression level.

Our approach modifies the contents of the neurons' memory. It can even create a new entry that was not part of the learning process. We see this property 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.

IV. Experiment and Results

In order to validate our work, we used our framework to compress the memory of a VG-RAM WNN system designed for traffic sign recognition [10]. In the next subsections, we briefly present the traffic sign recognition system (Section IV-A). Then, we describe the dataset used for our experiments (Section IV-B). Thereafter, we detail the experiments and the results obtained with our compression framework to enable fast response time applications on standard computers (Section IV-C) and acceptable interactive times on small low-power systems (Section IV-D).

A. Traffic Sign Recognition Application

The traffic sign application used in this paper employs a VG-RAM architecture with one neural layer connected to a (50 × 50)-sized input grayscale image (please see Fig. 3 for an overview of the system, and refer to [10] for details). The synaptic mapping function that maps non-binary image pixels to binary values is a *minchinton cell* [16] that works as follows. The non-binary value read by each synapse connected to the grayscale image is subtracted from the value read by the subsequent synapse in the set of synapses of each neuron, **S**, where $|\mathbf{S}| = p = 64$ (the last synapse, s_p, is subtracted from the first, s_1). If a negative value is obtained, the bit corresponding to that synapse is set to one. Otherwise, it is set to zero. The receptive field of each neuron is randomly defined during the network creation and follows a Normal distribution centered in the position of the neuron mapped to the input layer (see [10]). The centers of the receptive fields are mapped to the input according to a log-polar transform that mimics interconnection patterns observed in the biological vision system [10]. Subsequent use of the synapses assumes the same receptive field.

In contrast to the system proposed in [10], where one grayscale image was generated for each color channel for improving recognition performance, in this paper only the channel with the highest accuracy was chosen to run our experiments, i.e. the green channel. Additional experiments have shown that the compression method used here is not channel dependent (Section IV-C).

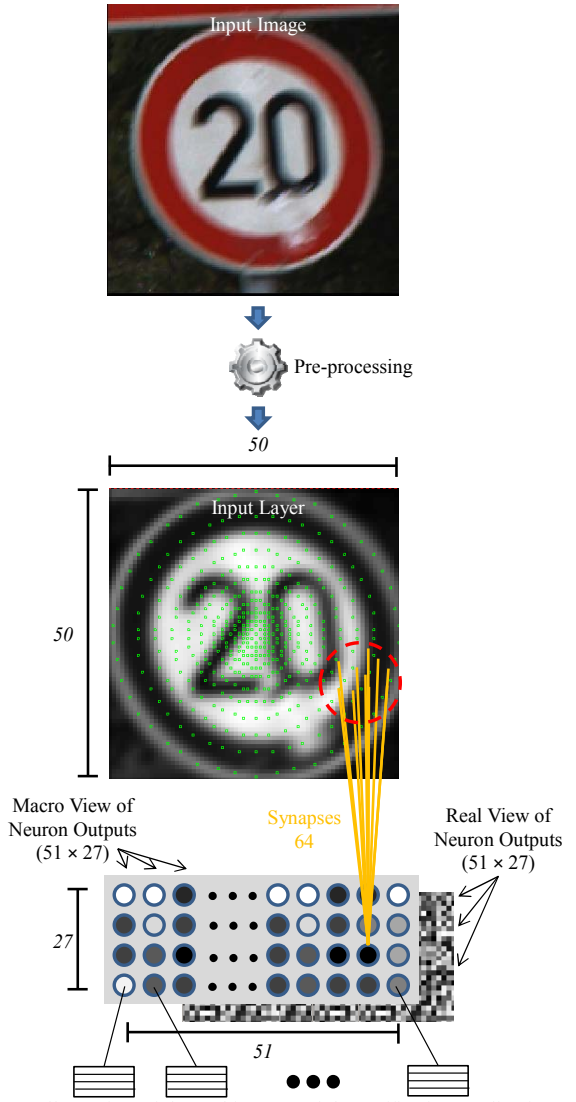


Fig. 3. Illustration of the architecture of the traffic sign application. It receives a colored input image that is pre-processed (cropped, scaled, filtered, etc.). The resulting grey scale image is set as the Input Layer of a single layer VG-RAM. The Neural Layer, comprising 51 × 27 neurons, samples the Input Layer according to a log-polar mapping, i.e. the center of the Receptive Field of each neuron maps to the Input Layer in a log-polar fashion (green dots). Each neuron samples the input layer with 64 Synapses that are randomly distributed within its Receptive Field (red traced circle) according to a Normal distribution. This figure illustrates a macro view of the neurons (front) in addition to an example of the real outputs of the neurons (back).

B. Dataset

The experiments in this work follow the same setup described in [10] and, therefore, use the German Traffic Sign Recognition Benchmark (GTRSB) dataset (http://benchmark.ini.rub.de/) [17, 18]. The GTRSB comprises 39,209 training images and 12,630 test images for $l = 43$ different classes of traffic signs. Each image is cropped on a specific traffic sign and has a size between 15 × 15 and 250 × 250 pixels. Examples of traffic sign images from the GTRSB database can be seen in Fig. 4.

All traffic sign images of the dataset were rescaled to 50 × 50 pixels in order to fit the VG-RAM input size.

Fig. 4. Examples of traffic sign images of the GTRSB database.

C. Desktop Experiments

Using the training and test framework proposed in [17], we first trained a VG-RAM WNN with 39,209 images. To compensate for small annotation errors in the dataset, each image was trained three times: one with the original image, and two with small random variations in scale, orientation and translation. The resulting memory (input-output memory lines) of each neuron, considering a neural layer with $n = 1{,}377$ neurons [10], was compressed according to the framework described in Section III. With the resulting compressed memory, we performed a variety of experiments to examine the performance of our framework, i.e. the tradeoff between reducing network's memory footprint and system runtime, while maintaining high and acceptable classification accuracy.

Our first experiment analyses the behavior of the compressed memory of each neuron using our proposed framework. Initially, each neuron memory contained $m = 3 \times 39{,}209 = 117{,}627$ lines, equivalent to $(p / 8 + \text{sizeof}(t)) \times m \times n =$ 1,854MB of binary data, where the function sizeof(j) returns the number of bytes of the scalar j. These data were reduced with 4 compression levels: $cl = 10\%$ of the original data ($m = 11{,}762$ lines – around 185MB), $cl = 1\%$ of the original data ($m = 1{,}160$ lines – around 18MB), $cl = 0.5\%$ ($m = 570$ lines – around 9MB) and $cl = 0.25\%$ ($m = 272$ lines – around 4MB). Please note that 0.25% is the lowest possible level in order to maintain the memory balance and avoid erasing a traffic sign type from the dataset. The relation between compression level and memory footprint is illustrated in Fig. 5.

As Fig. 5 shows, the memory footprint (obviously) decreases linearly with the number of lines in the Look-up Table of each neuron. Therefore, by decreasing the number of lines, our framework is able to reduce the memory footprint of the application, which is particularly important for applications running on desktop computers with limited memory or on small low-power systems.

Our second experiment analyses the effect of the memory compression in the runtime of the overall system. For this experiment, we ran our C/C++ OpenMP optimized code 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. We ran the system with all five compression levels (original memory, and memory with compression levels 10%, 1%, 0.5% and 0.25%) for all 12,630 test images in the dataset. The runtime per image was calculated as an average over all images in the test dataset. The relation between compression level and system's runtime is shown in Fig. 6.

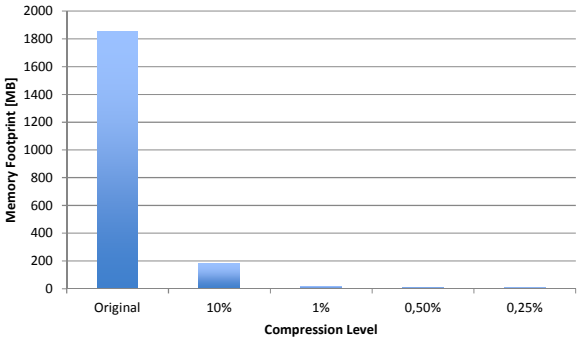


Fig. 5. Memory footprint of our traffic sign recognition application for several memory compression levels, *cl*.

As Fig. 6 shows, there is an almost linear relationship between compression level and system's runtime. For example, running the system with the original memory takes in average around 470ms to process each image. This value is reduced to around 50ms for the first level of compression (10% of the original memory). Other levels of compression allow processing each input image even faster (for example, 8ms per image with 1% of the original memory). This result shows that our compression framework enables very fast implementations of our VG-RAM based system.

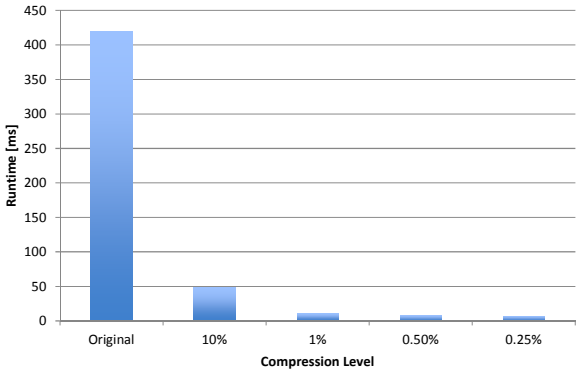


Fig. 6. Runtime for our traffic sign recognition application using VG-RAM WNN. We compared the system's runtime using the original memory and all four compression levels.

Our third experiment analyses the effect of the memory compression in the classification performance of the system. We ran the system with all five possibilities (original memory, and memory created with compression levels 10%, 1%, 0.5% and 0.25%) for all 12,630 test images in the dataset. The relation between compression level and system's classification performance is shown in Fig. 7.

As shows in Fig. 7, we can see a small reduction in accuracy as the compression level increases. As an example, the classification accuracy of the system with the original memory is around 97.93%. This value is reduced to around 97.42% for the first level of compression (10% of the original memory). Reducing the memory even further (1% of the original memory) decreases the precision to around 96.11%, which is still high and acceptable for such complex

application. This result shows that our compression framework is able to create compressed memories that maintain a high and acceptable classification performance for our VG-RAM based system.

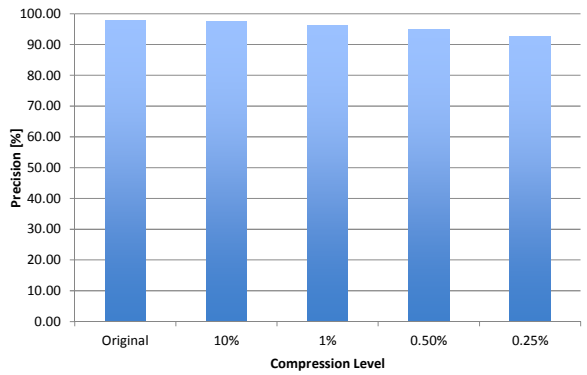


Fig. 7. Classification performance for our traffic sign recognition application using VG-RAM WNN. We compared the system's accuracy using the original memory and all four levels of compression.

We also performed one experiment to compare the classification performance of our framework against random deletion of neuron memory entries. As shown in Fig. 8, for small memory compressions, the difference between our framework and random deletion is small. For example, the classification accuracy using the first level of compression (10% of the original memory) is around 97.42%, against 96.98% for random deletion. This demonstrates that the VG-RAM architecture creates many redundant entries in the memory, and even randomly eliminating entries will not affect much the classification performance. However, as the compression level increases, we can see a gap increasing between the performance of our framework and random deletion. For example, the classification accuracy of our framework using the last level of compression (0.25% of the original memory) is around 92.73%, against 89.8% for random deletion. This demonstrates that a careful scheme of eliminating entries of the neurons' memory is useful and improves the overall accuracy.

Our next experiment verifies if our compression technique is color-channel dependent. The original system presented in [10] used a voting scheme with a combination of all three image color channels, red, green, and blue, to improve the overall classification performance. This comes with an increase in running time and memory footprint. As we would like to reduce these costs, we decided to run all experiments in this paper using the channel with the highest accuracy, i.e. the green channel. The behavior of the classification performance of our compression framework for all color channels is shown in Fig. 9.

In Fig. 9, we can see similar performances using the original memory for all color channels, with the green channel performing slightly better than the others do. The system accuracy using the second level of compression (1% of the original memory) also shows similar small reductions in classification performance, demonstrating that our framework is not channel dependent and could potentially be applied to other applications.

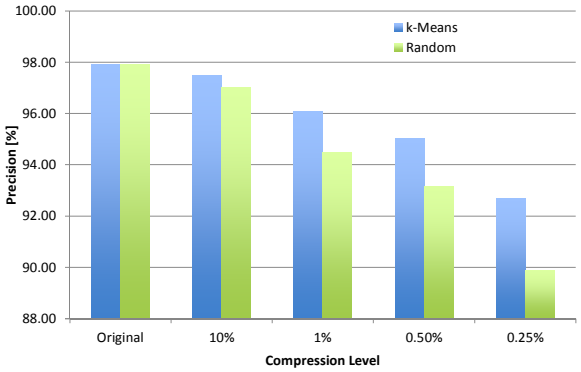


Fig. 8. Classification performance for our traffic sign recognition application using VG-RAM WNN. As shown in the figure, carefully compressing the memory using our framework slightly affects the overall classification performance. In contrast, random deletion of neuron memory entries creates a more significant decrease in classification performance.

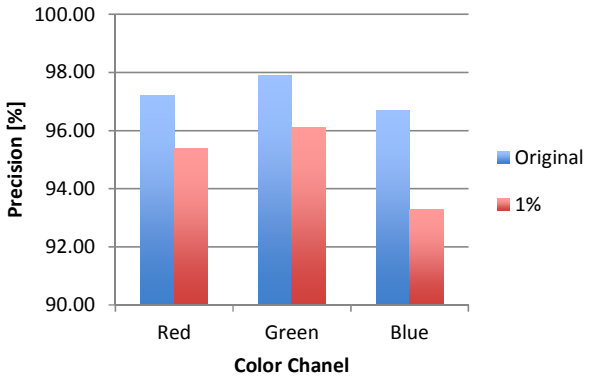


Fig. 9. Classification performance for our traffic sign recognition application using VG-RAM for the different color channels, red, green, and blue. Our framework shows similar classification results for all three color-channels, with the green channel performing slightly better than the others using the original and the compress memory.

D. *Experiments with Small Low Power Systems*

Considering the advances in mobile processing, small low power, and embedded systems, we used the same C/C++ code (i.e. not optimized for special embedded environments) to verify the potential of our compression framework in two simple and cheap devices: RaspBerry Pi Model B revision 2[1] and pcDuino V1[2]. Our experiments show that it is possible to run a VG-RAM traffic sign recognition system on such devices using our memory-compressing framework.

It is clear that such an application with full memory would be impossible to be processed or even loaded on most embedded systems because of their limited memory and processing power. Therefore, we decided to test our system on the two mentioned low power systems with four levels of compression: cl = 5% of the original data (m = 5,881 lines –

[1] http://www.raspberrypi.org
[2] http://www.pcduino.com

around 92,7MB), cl = 1% of the original data (m = 1,176 lines – around 19MB), cl = 0.5% (m = 588 lines – around 9MB) and cl = 0.25% (m = 294 lines – around 5MB). Please note that we started with the compression level of 5% in order to compare the results of both devices, since the RaspBerry Pi cannot load our complete VG-RAM traffic sign architecture with compression level 10% on its memory.

First, we used a pcDuino device, equipped with a 1 GHz ARM Cortex A8 processor and 2GB of RAM running Linux. We ran the system with all four possibilities (memory created with compression levels 5%, 1%, 0.5% and 0.25%) for a subset of the test images in the dataset. The runtime per image was calculated as an average over all tested images. The relation between compression level and system runtime is shown in Fig. 10. As shown in this figure, our pcDuino system is able to process the images at interactive rates (i.e. around 4-6 Hz) for different levels of memory compression.

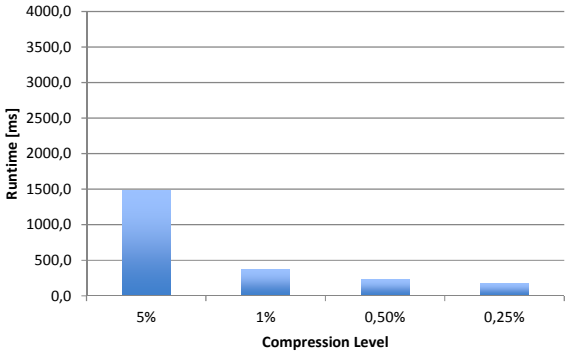


Fig. 10. Runtime for our traffic sign recognition application using VG-RAM WNN running on the pcDuino device. Interactive rates can be achieved with memory compression.

We also tested a RaspBerry Pi device, equipped with a 700MHz processor and 512MB of RAM running Linux. As before, we ran the system with all four possibilities for a subset of the test images in the dataset. The relation between compression level and system runtime is shown in Fig. 11. As shown in this figure, our RaspBerry Pi system needs more time to process the images than the pcDuino. However, it can be seen that with the use of memory compression, the traffic sign recognition application running on a RaspBerry Pi achieves similar runtime to the same application without memory compression running on a standard PC (please compare the last two bars of the graph of Fig. 11 with the first of the graph of Fig. 6). This illustrates the advantage of our clustering scheme.

V. Conclusions and Future Work

In this paper, we presented a new framework for applying VG-RAM WNN to complex multi-class classification problems. We have shown that by using clustering techniques, it is possible to eliminate irrelevant and redundant information from the input-output Look-up Table of each neuron with little compromise of the classification performance. As a result, our framework is able to reduce the network memory footprint and its runtime enabling its application in systems that require very fast response times in standard computers, and at interactive rates for small low-power systems. The classification performance is slightly reduced depending on the level of memory compression. However, this can be justified by a gain in speed and in memory usage.

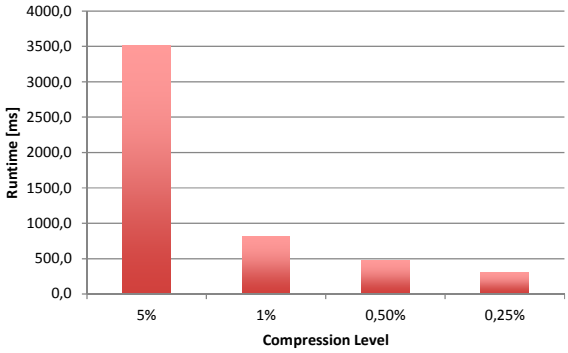


Fig. 11. Runtime for our traffic sign recognition application using VG-RAM WNN running on a RaspBerry Pi.

We evaluated the performance of our framework using the GTSRB dataset, creating a traffic sign recognition system based on VG-RAM WNN. Our experiments showed that our system can be employed for traffic sign recognition with good accuracy (recognition rates around 95%) at 121.3 recognitions per second in standard computers with a 0.5% memory compression level. Using the same setup, our system could run on a PcDuino at 4.3 recognitions/s and on a RaspBerry Pi at 2.1 recognitions/s, requiring a memory footprint of only 9MB to store the neurons' memory.

One of the main advantages of VG-RAM over other neural network approaches is its simple implementation and fast training. We believe that the memory compression framework proposed in this paper can help improving the test performance of VG-RAM, especially for applications requiring large training datasets.

For future work, we would like to evaluate the performance of other clustering techniques for memory compression and evaluate the overall system on different multi-class classification problems.

References

[1] I. Aleksander, "Self-adaptive universal logic circuits," *Electronics Letters*, vol. 2, no. 8, pp. 321–322, Aug. 1966.

[2] I. Aleksander, "From WISARD to MAGNUS: A Family of Weightless Virtual Neural Machines," in *RAM-Based Neural Networks*, Singapore: World Scientific Publishing Co Pte Ltd, 1998, pp. 18–30.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.