Memory Efficient Weightless Neural Network using Bloom Filter

Leandro Santiago[1], Leticia Verona[2], Fabio Rangel[2], Fabrício Firmino[2], Daniel S. Menasche[2], Wouter Caarls[3], Mauricio Breternitz Jr[4], Sandip Kundu[5], Priscila M.V. Lima[2] and Felipe M. G. França[1] *

1- PESC/COPPE 2- NCE
Universidade Federal do Rio de Janeiro, Brazil
3- PUC-RIO, Brazil 4- IST & INESC-ID, University of Lisbon, Portugal
5- Department of Electrical and Computer Engineering,
University of Massachusetts Amherst, USA

Abstract. Weightless Neural Networks are Artificial Neural Networks based on RAM memory broadly explored as solution for pattern recognition applications. Due to its memory approach, it can easily be implemented in hardware and software providing efficient learning mechanism. Unfortunately, the straightforward implementation requires a large amount of memory resources making its adoption impracticable on memory constraint systems. In this paper, we propose a new model of Weightless Neural Network which utilizes Bloom Filters to implement RAM nodes. By using Bloom Filters, the memory resources are widely reduced allowing false positives entries. The experiment results show that our model using Bloom Filters achieves competitive accuracy, training time and testing time, consuming up to 6 order of magnitude less memory resources in comparison with the standard Weightless Neural Network model.

1 Introduction

Weightless Neural Networks (WNNs) [1] are neuron models based on Random Access Memory (RAM) where each neuron is defined as RAM node. These models have been shown as attractive solution to solve pattern recognition and artificial consciousness applications achieving surprisingly performance. WiSARD (Wilkie, Stoneham and Aleksander's Recognition Device) is the pioneering WNN distributed commercially [2] which provides simple and efficient implementation enabling to deploy learning capabilities into real-time and embedded systems.

However, the straightforward WiSARD implementation requires a considerable amount of memory resources to obtain good learning features. To address this problem, we propose a new WiSARD model that replaces RAM nodes to Bloom Filters. Bloom Filters [3] are probabilistic data structures which represent a set as small bit array allowing the occurrences of false positive. Our experiments analyze accuracy, training time, testing time and memory consumption of our model compared to standard WiSARD and WiSARD implemented with Hash Table.

*The authors would like to thank CAPES, CNPq and FAPERJ for the financial support of this work.

2 Background

2.1 WiSARD

WiSARD (Wilkie, Stoneham and Aleksander's Recognition Device) is a multi-discriminator WNN model proposed in the early 80's [2] that recognizes patterns from binary data. Each class is represented by a discriminator which contains a set of RAMs with one-bit word to store the relevant knowledge, so that class prediction can successfully be performed. Before sending the input data to the discriminators, they need to be converted in a binary format using some transformation which depends on the data type. The binary input with $N \times M$ bits is splitted in N tuples of M bits, where each tuple is a memory address to access one of the N RAMs with 2^M locations. A pseudo-random mapping connects the tuples to the binary input and each discriminator has its own pseudo-random mapping.

Initially, all RAMs have zero (0) in their locations. On the training phase, the training input is sent to the related discriminator which sets to one (1) all accessed RAM positions. On the classification phase, the input is sent to all discriminators generating responses per discriminators by summing all accessed RAM values from each one. The discriminator with the highest response is chosen as representative class of the input. In both phases, the pseudo-random mapping from input to the tuples is the same for each discriminator.

2.2 Bloom Filter

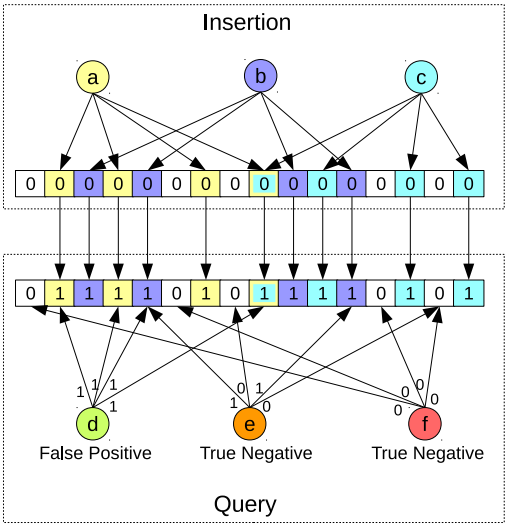

Fig. 1: Bloom filter operations example with 16-bit array and 4 hash functions.

Bloom Filters [3] are space-efficient data structure for Approximate Membership Query (AMQ) which test whether an element belongs to a given set or not with a certain false positive probability. In other words, sometimes the membership query will respond that an element was stored while actually it was not inserted. Bloom Filter is composed of a m-bit array and k independent hash functions that map an element into k bit array positions.

The standard Bloom Filter supports insertion and query operations as exemplified in Figure 1. Initially, all bit array positions are zeroed. In the insertion operation, an element is mapped into k positions where each one is set to 1. In the example, a, b and c are inserted using 4 hash functions. The query operation lookups the k positions mapped from the input element, indicating it as either a member of set considering a false positive rate if all values are $1's$ or a non-member when any value is 0. In Figure 1, d is a false positive since it was suggested as member of set (only a, b and c were inserted), while e and f do not belong to the set. Note that Bloom Filter correctly reports true negative whenever an element is not a member itself.

The probability of false positive p is affected by the parameters m, n and k, corresponding to bit array size, number of elements to store and number of hash functions, respectively [4]. Given the probability p and capacity n, it is possible to determine the ideal parameters m and k. The number of bits m is calculated by the Formula (1) [5], while the number of hash functions k is obtained by the Formula (2) [4].

$$m = -n \times \frac{\ln(p)}{\ln(2)^2} \tag{1}$$

$$k = m \times \frac{\ln(2)}{n} \tag{2}$$

3 WiSARD based on Bloom Filters

In WiSARD, the binary transformation impacts in the accuracy and the learning capacity of the model affecting straightforwardly its input size, that determines the number of RAMs and the tuple size for each discriminator. Thus, huge RAMs might be required to achieve a good accuracy.

After training big RAMs, it is common few memory addresses are accessed in comparison with the total positions resulting in several positions with $0's$. We extend WiSARD by replacing RAMs to Bloom Filters in order to reduce its memory resources avoiding it stores irrelevant zero positions. The new model is termed Bloom WiSARD. The key idea is to store a set of tuples mapped to each RAM and tests if a given tuple belongs to its corresponding set.

Figure 2 presents the Bloom WiSARD design. On the training phase, the tuples are inserted into Bloom Filters updating the k bit array positions. Whereas on the classification phase, the tuples are queried into their associated Bloom Filters returning whether each tuple is a member or not by ANDing all k bit values. In the similar way the WiSARD, the discriminator responses are calcu-

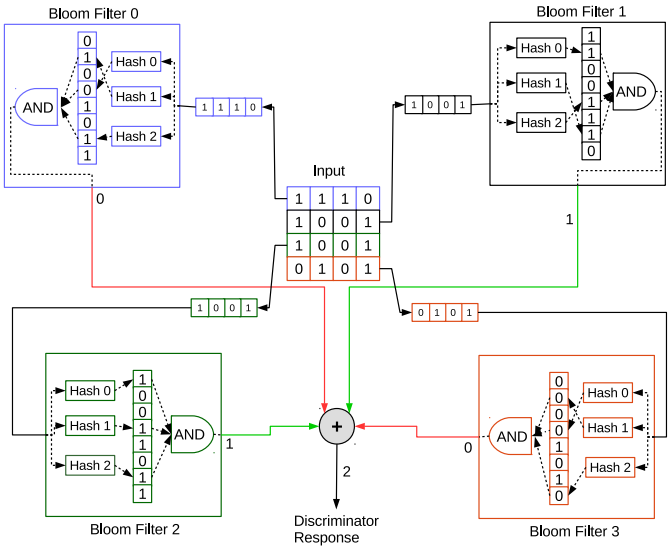

Fig. 2: Bloom WiSARD design.

lated by summing the N Bloom Filter membership results so that the highest response selects the appropriate discriminator to represent the input.

Our Bloom WiSARD implementation utilizes double hashing technique [6] to generate k hash functions in the form: $h(i,k) = (h_1(k) + i \times h_2(k)) \pmod{n}$, where h_1 and h_2 are universal hash functions. We adopt MurmurHash function[7] as seed hashes h_1 and h_2.

4 Experiments and Results

4.1 Dataset

To evaluate our proposed model, we compare Bloom WiSARD to two different WiSARD versions: standard WiSARD and dictionary WiSARD that implements each RAM as Hash Table storing key-value pair in each position where the key is the memory address. We select the MNIST database [8] and a subset of binary classification and multiclass classification datasets used in [9]. The most of the problems were taken from UCI public repository [10] and they have different characteristics in term of number of samples, number of classes and number of features. Some datasets do not provide the training set and testing set in separated files. For these datasets, we adopt the same methodology applied in [9]: we random shuffle the single data and partition it in 3 parts such that 2/3 are used as training set and 1/3 compose the testing set.

Table 1: Accuracy, training time, testing time and memory results for Binary Classification problems.

Dataset	WNN	Accuracy	Training (s)	Testing (s)	Memory (KB)
Adult	WiSARD	0.722	4.414	1.05	8978432
	Dict WiSARD	0.721	1.947	1.188	383.535
	Bloom WiSARD	0.718	1.932	1.166	80.173
Australian	WiSARD	0.843	0.002	0.001	4096
	Dict WiSARD	0.841	0.002	0.001	11.299
	Bloom WiSARD	0.834	0.002	0.001	8.613
Banana	WiSARD	0.87	0.052	0.028	13312
	Dict WiSARD	0.871	0.054	0.033	23.428
	Bloom WiSARD	0.864	0.058	0.036	3.047
Diabetes	WiSARD	0.698	0.001	0.0007	2048
	Dict WiSARD	0.689	0.001	0.0008	6.553
	Bloom WiSARD	0.69	0.001	0.0008	4.793
Liver	WiSARD	0.593	0.001	0.0007	5120
	Dict WiSARD	0.587	0.001	0.0008	6.387
	Bloom WiSARD	0.591	0.001	0.0009	2.344

4.2 Experimental Setup

The experiments were performed on CPU machine that has an Intel Core i7-6700(3.40GHz) processor with 32GB of RAM running Ubuntu Linux 16.04 operating system. All WiSARD versions were implemented using C++11 language with single thread providing a library to be utilized in Python, while the experiments were implemented in Python. To convert the input attributes to binary format, we concatenate all binary attributes using thermometer to transform the continuous attributes and one hot encoding to transform categorical attributes. The input size, number of RAMs and tuple size varying according to dataset, however these parameters are kept to all WiSARD versions. Bloom filters are setup with 10% of false positive probability, the capacity are empirically selected from each dataset and m and k are calculated through the formulas presented in Section 2.2.

4.3 Results and Discussion

All results are obtained through the mean of 20 runs with negligible standard deviation. Tables 1 and 2 show the results for binary classification and multiclass classification datasets, respectively. Overall, Bloom WiSARD achieved comparable accuracy, training time and testing time results while consuming a smaller amount of memory than other versions. The memory consumption is reduced up to 6 order of magnitude compared to standard WiSARD and approximatelly 7.7 times when compared with dictionary WiSARD.

Table 2: Accuracy, training time, testing time and memory results for Multiclass Classification problems..

Dataset	WNN	Accuracy	Training (s)	Testing (s)	Memory (KB)
Ecoli	WiSARD	0.793	0.0005	0.0005	7168
	Dict WiSARD	0.799	0.0005	0.0005	5.664
	Bloom WiSARD	0.799	0.0005	0.0007	3.281
Iris	WiSARD	0.985	0.0001	0.000009	1536
	Dict WiSARD	0.977	0.0001	0.000008	0.747
	Bloom WiSARD	0.976	0.0001	0.0001	0.703
Letter	WiSARD	0.845	1.483	0.16	10223616
	Dict WiSARD	0.846	0.0717	0.22	121.748
	Bloom WiSARD	0.848	0.07	0.208	91.292
MNIST	WiSARD	0.917	4.317	0.33	9175040
	Dict WiSARD	0.916	0.811	0.475	1368.457
	Bloom WiSARD	0.915	0.77	0.369	819.049
Satimage	WiSARD	0.851	0.048	0.034	27648
	Dict WiSARD	0.853	0.05	0.049	69.141
	Bloom WiSARD	0.851	0.053	0.05	12.656
Shuttle	WiSARD	0.87	0.119	0.064	8064
	Dict WiSARD	0.869	0.12	0.078	4.956
	Bloom WiSARD	0.868	0.132	0.103	3.691
Vehicle	WiSARD	0.67	0.003	0.0021	9216
	Dict WiSARD	0.672	0.003	0.0026	17.617
	Bloom WiSARD	0.662	0.003	0.0028	4.219
Vowel	WiSARD	0.876	0.0023	0.0025	14080
	Dict WiSARD	0.876	0.0023	0.0032	16.221
	Bloom WiSARD	0.876	0.0022	0.0036	6.445
Wine	WiSARD	0.932	0.0006	0.0003	4992
	Dict WiSARD	0.924	0.0005	0.0003	4.248
	Bloom WiSARD	0.926	0.0005	0.0004	2.285

5 Conclusion

WiSARD is an efficient WNN model based on RAM memory that can be trivially implemented in hardware and real-time systems. Nevertheless, some applications require a huge amount of memory to achieve good learning capabilities. In this work we propose Bloom WiSARD model which implements RAM nodes as Bloom Filters. By using Bloom Filters, the memory resources is widely reduced with tradeoff of occurrences of false positive. Our experiments show that the model provide good accuracy, training and testing time results. In addition, it consumes up to 6 order of magnitude less resources than standard WiSARD and about 7.7 times less resources than WiSARD implemented with dictionaries. Future work will focus to extend Bloom Filter operations such as counts

frequency of elements stored in order to enable Bloom WiSARD to use improvement techniques as DRASiW.

References

[1] Igor Aleksander, Massimo De Gregorio, Felipe Maia Galvão França, Priscila Machado Vieira Lima, and Helen Morton. A brief introduction to weightless neural systems. In *ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22-24, 2009, Proceedings*, 2009.

[2] I. Aleksander, W.V. Thomas, and P.A. Bowden. WisardÂ·a radical step forward in image recognition. *Sensor Review*, 4(3):120–124, 1984.

[3] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[4] Peter C. Dillinger and Panagiotis Manolios. Bloom filters in probabilistic verification. In Alan J. Hu and Andrew K. Martin, editors, *Formal Methods in Computer-Aided Design*, pages 367–381, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[5] P. Sterne. Efficient and robust associative memory from a generalized bloom filter. *Biological Cybernetics*, 106(4):271–281, Jul 2012.

[6] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: Building a better bloom filter. In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006*, pages 456–467, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Murmurhash fuction. https://en.wikipedia.org/wiki/MurmurHash.

[8] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[9] G. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, April 2012.

[10] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.