ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

Efficient Knowledge Aggregation Methods for Weightless Neural Networks

Otávio Oliveira Napoli[1],[2], Ana Maria de Almeida[1], José Miguel Sales Dias[1], Luís Brás Rosário[3], Edson Borin[2] and Mauricio Breternitz Jr[1] *

1- Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, Lisbon, Portugal

2- Institute of Computing – UNICAMP, Campinas, São Paulo, Brazil

3- Faculty of Medicine, Lisbon University, Lisbon, Portugal

Abstract. Weightless Neural Networks (WNN) are good candidates for Federated Learning scenarios due to their robustness and computational lightness. In this work, we show that it is possible to aggregate the knowledge of multiple WNNs using more compact data structures, such as Bloom Filters, to reduce the amount of data transferred between devices. Finally, we explore variations of Bloom Filters and found that a particular data-structure, the Count-Min Sketch (CMS), is a good candidate for aggregation. Costing at most 3% of accuracy, CMS can be up to 3x smaller when compared to previous approaches, specially for large datasets.

1 Introduction

Weightless Artificial Neural Networks (WNNs) [1] have a high discriminative capacity along with reduced computational load, which makes them ideally suited for applications in which multiple devices connected through the internet collaborate in distributed, semi-autonomous ways, for monitoring and control tasks driven by machine learning algorithms. The robustness, simplicity, and light computational load of WNNs make them an ideal model for different scenarios such as IoT, open-set recognition, and use in constrained, and embedded devices.

The above factors also make WNNs a good candidate for Federated Learning (FL) scenarios. FL is decentralized approach to training machine learning models that allows multiple devices to collaboratively train a global model without exchanging user data. Each device trains a local model using only local data, and the local models are then aggregated into a global model. The aggregation of the local models can be performed in different ways, such as averaging the weights of the models, or by using a weighted average [2].

In this work we show how to aggregate the knowledge of multiple WNNs into a single, global, model. We also show how to aggregate more compact data structures used in WNNs, such as Bloom Filters [3], to reduce the amount of data transferred between devices. Finally, we explore variations of Bloom Filters and show that a particular data-structure, the Count-Min Sketch (CMS), is a

*This work was partially supported by Fundação para a Ciência e a Tecnologia, I.P. (FCT) [ISTAR Projects: UIDB/04466/2020 and UIDP/04466/2020] and DSAIPA/AI/0122/2020 Aim Health. The authors would also like to thank CNPq (314645/2020-9 and 404087/2021-3) and Fapesp (2013/08293-7) for their financial support, and Discovery lab for its technical support.

good candidate for this task. Costing at most 3% of accuracy, CMS can be up to 3x smaller when compared to previous approaches.

2 Weightless Neural Networks

A n-tuple classifier [1] is a binary pattern classifier composed by random access memories (RAMs) R, where each RAM is a one-bit vector of size 2^n that can represent all binary patterns of size n, the addresses. The classifier can be trained to learn the binary patterns of size n (known as tuple size) from a training set.

The training process is illustrated in Figure 1. The training set is a collection of samples, *i.e.*, binary patterns of size m. The training process consists in iterating once over all samples of the training set. Each sample is divided into $\frac{m}{n}$ patterns of size n, based on a random mapping M. Then, each pattern will become an address that will be used to access the respective RAM and record the presence of that pattern. Once trained, given a sample, the similarity score of this sample can be calculated by counting the number of patterns presented in the respective RAMs. This score is usually used to discriminate the class of the sample.

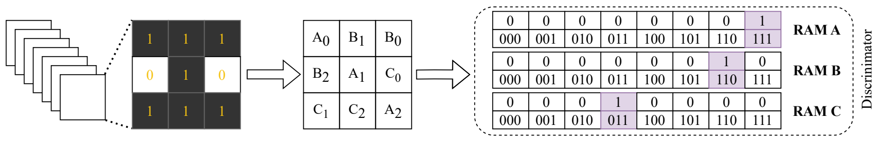


Fig. 1: Training a n-tuple classifier model.

WiSARD [4] is a n-Tuple classifier composed of several RAM-discriminators, one for each class to be discriminated. Unlike traditional n-tuple classifiers, a WiSARD is composed by a set of D discriminators, where each discriminator has a set of $\frac{m}{n}$ RAMs. A single-discriminator WiSARD (which is equivalent to n-Tuple classifier) is illustrated in Figure 1. Each discriminator is trained with samples from a specific class, as each discriminator aims to identify that class samples. The inference of the WiSARD model is performed by calculating the similarity score of the sample for each discriminator, and the class with the highest similarity score is the predicted class.

As the number of samples of the training dataset grows, the WiSARD model's RAMs may suffer from saturation, especially when the dataset presents many outliers. DRASiW [5] is an extension of WiSARDs that stores the frequency of observations of a given pattern into the RAM, instead of just relying on a presence bit. Thus, the RAM ceases to be a one-bit vector and becomes a vector of frequency counters. DRASiW allows the filtering of noisy patterns by using a technique called Bleaching [6], where patterns that occur less than a threshold are considered too uncommon and do not contribute to the discriminator's response. In this work, we use the DRASiW model, where RAMs are vectors of integers, being more robust to noisy data.

3 Efficient Aggregation Methods for WiSARD models

Considering a scenario with several homogeneous WNNs [1] models, the knowledge aggregation can be done by combining the corresponding RAM's contents. As each RAM cell counts the number of occurrences (*i.e.*, frequency) of a given pattern, the models' knowledge can be aggregated by adding the frequencies of the patterns in each model. This is called *frequency aggregation*.

However, even though the aggregation function itself is not computationally expensive, in terms of data transfer cost, a naive approach requires transferring the contents of all RAMs. In fact, the amount of data that needs to be transferred is the same as the model's size. Hence, the transfer cost (in bytes) is $T_{cost} = D \cdot R \cdot C \cdot 2^n$, where C is the number of bytes used to represent the integer value, D is the number of discriminators, and R is the number of RAMs. To deal with those (potentially huge) amounts of data, RAMs are commonly implemented using dictionary/hash table structures. This approach is called Dict-WiSARD and can greatly reduce the amount of data stored. However, even so, in the worst case this approach requires $\mathcal{O}(2^n)$ bytes to be transferred.

Da Silva *et al.* [7] were the first to explore the use of Dict-WiSARD to perform collaborative learning. Despite the reduced memory consumption, Santiago *et al.* [3] have shown that Dict-WiSARDs still consume a huge amount of memory when compared to other data structures, such as Bloom Filters, leading to increased data transference cost.

3.1 Bloom Filters as Space-Efficient Data Structures

Santiago *et al.* [3] suggest that Bloom Filters (BF) can be applied to improve the design of discriminators, allowing one to explore a wide range of solutions trading between memory costs and classifier's performance. BFs are space-efficient probabilistic data structures used for membership queries. The false positive rate (FPR) of the BF, *i.e.*, the probability of obtaining a wrong answer, depends on the number of different hash functions, the available positions and the number of elements added [2]. In terms of space occupation, BF are linear on the number of available positions. However, to implement DRASiW systems, variations of BFs that allow counting should be used [3].

Counting Bloom Filter [8] (CB) is similar to a standard BF except that it uses a frequency vector and therefore the CB can be used to estimate the number of insertions of a given element. Thus, it can be used as RAMs to return the maximum number of observations of a given address. The size of CB is the same as the BF, but multiplied by a constant, which is the size of the integer used to represent the frequency.

Count-Min Sketch [9] (CMS) is another probabilistic data structure used to track how many times an element appeared in a data stream. Similarly to CB, CMS attempts to approximate the frequency of an element. Instead of having

[1]Models that share the same parameters (M and n) and perform the same task

[2]In fact, the filter size is inversely proportional to the desired FPR, meaning that increasing the filter size reduces the FPR while allowing for more elements to be stored.

a single frequency vector of size W as CB, where the element is hashed with k hash functions, CMS adds a dimension, d (depth), based on the number of hash functions. Thus, the data structure is a matrix of size $W \cdot d$.

One interesting property of BFs is that they can be joined [10]. Given two BFs that share the same parameters, *e.g.*, B_1 and B_2, the union of both results is a new BF $B_3 = B_1 \cup B_2$ containing the elements present in both sets. In the context of WNNs, BFs are not only space-efficient data structures to design RAMs [3] but are also good structures to perform knowledge aggregation, as they support the union operation[3]. It is worth noticing that the choice of the data-structure, as well as its parameters, will impact not only on the space complexity of the aggregation, but also on the performance. The next sections present experiments used to evaluate these aspects when aggregating WNNs.

4 Experimental Setup

We evaluated the memory footprint and performance of the aggregated model for each data-structure by experimenting with different datasets and parameters. Each experiment consists of training 4 homogeneous WNNs individually, each using a 25% partition of the dataset. Then the models are aggregated into a single model, which is evaluated. Each experiment was run three times with different partitions of the dataset, and the results were averaged. We used the public datasets described in Table 1. Most coming from the UCI repository[4], plus MNIST[5], and MotionSense[6]. A specially designed library of modules, the `wisardlib` package, was implemented, being the first pure python package that implements scikit-learn interfaces for training and evaluating WNN models[7].

Dataset	Size (KB)	Classes	#Train	#Test	Balanced	Performance	Memory (KB)
breast cancer	144	2	398	171	no	0.95	9.58
glass	24	6	149	65	no	0.64	12.40
iris	12	3	105	45	yes	0.96	1.06
letter	2664	26	1400	6000	yes	0.86	301.63
mnist	53672	10	60000	10000	yes	0.91	8975.11
motionsense	12512	6	3414	1020	yes	0.63	1448.64
satimage	1864	6	4501	1929	no	0.87	108.21
vehicle	132	3	676	170	yes	0.69	5.57
wine	24	3	124	54	no	0.96	3.11

Table 1: Description of the datasets used in the experiments, the performance (f1-score or accuracy) and memory footprint using Dict-WiSARD (baseline).

Since the datasets contain real-valued data, a preprocessing step converts each sample into an array of binary values. Table 2 shows the encoder used and the resolution for each of the datasets. The thermometer encoding is a way to represent a number using a binary code where each bit position corresponds to a value, and the bit is set to 1 if the value is less than a threshold. The resolution is the size of binary code. In all the cases where thermometer encoding did not

[3]CB and CMS data structures also support the union operation
[4]`https://archive.ics.uci.edu/ml/index.php`
[5]`http://yann.lecun.com/exdb/mnist/`
[6]`https://github.com/mmalekzadeh/motion-sense`
[7]`https://github.com/otavioon/wisardlib/`

Dataset	Encoder	Resolution	n	Bleaches
breast cancer	Dist. Thermometer	16	16	2, 5, 10
glass	Dist. Thermometer	64	24	3, 5, 10, 15, 20
iris	Dist. Thermometer	20	16	2, 5, 8, 10, 15
letter	Dist. Thermometer	14	16	30
mnist	Dist. Thermometer	16	16	40
motion sense	Thermometer	32	32	5, 10, 20
satimage	Thermometer	20	20	5
segment	Dist. Thermometer	16	16	15, 20, 25, 30, 40
vehicle	Dist. Thermometer	16	16	2, 5, 8, 10, 15
wine	Dist. Thermometer	16	16	2, 5, 8, 10, 15

Table 2: Parameters used for each experiment.

achieve good results, the distributive thermometer was used. The distributive thermometer [11] is a variant of the thermometer encoder, where the data is split into percentiles of same probability and then encoded. The tuple size (n) was defined based on the work of Santiago *et al.* [3], as well as the values for the encoder and the resolution, that were then tuned experimentally. As saturation may affect the classification results, we experimented with different values of bleach [8]. Finally, for CB, the false positive rate (FPR) was set to 0.02, 0.05, and 0.08 based on the findings reported by Santiago *et al.* [3]. For CMS, the width (W) was set to 20, 50, and 100 and depth (d) of 2, 3, and 5. These values were chosen because they are close to the CB.

5 Experimental Results

Table 1 presents the performance and the memory footprint of the aggregated model using the traditional Dict-WiSARD implementations. These results are used as baseline, and follow the same approach as da Silva *et al.* [7]. Figure 2 presents the performance obtained with each configuration in relation to the baseline (dashed line). Values close to 1 show that the performance of the aggregated model is similar to the performance of the baseline. It is worth noticing that the performance of the aggregated model are similar to the baseline for almost all configurations when using the smaller datasets. For larger datasets (letter, motion_sense and MNIST) the performance loss is significant in some cases (up to 35%). This happens because probabilistic data-structures with smaller size are more likely to have more false positives, leading to higher performance losses when the number of elements inserted and their variety are high.

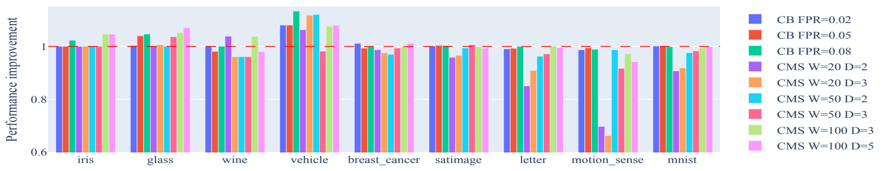


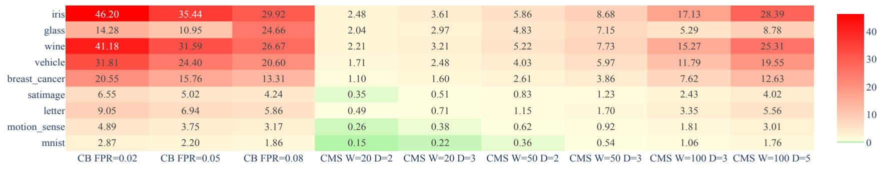
Fig. 2: Performance improvement in relation to baseline.

Figure 3 shows the ratio of the size of each configuration in relation to the baseline configuration, for each of the datasets. Some configurations are up to

[8]All results in the text are reported with the best parameters set. Besides that, for unbalanced datasets the f1-score was reported instead of accuracy.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

46× larger than the baseline. The CMS W=20 D=3 configuration is the one that produces the smallest memory footprint, but leads to a performance loss of 35%, making it unfeasible. There are 2 configurations CMS W=50 D=2 and CMS W=50 D=3 that lead to a performance loss of less than 3% and 2% in all datasets, with almost one third or one half of the memory footprint, respectively.



	CB FPR=0.02	CB FPR=0.05	CB FPR=0.08	CMS W=20 D=2	CMS W=20 D=3	CMS W=50 D=2	CMS W=50 D=3	CMS W=100 D=3	CMS W=100 D=5
iris	46.20	35.44	29.92	2.48	3.61	5.86	8.68	17.13	28.39
glass	14.28	10.95	24.66	2.04	2.97	4.83	7.15	5.29	8.78
wine	41.18	31.59	26.67	2.21	3.21	5.22	7.73	15.27	25.31
vehicle	31.81	24.40	20.60	1.71	2.48	4.03	5.97	11.79	19.55
breast_cancer	20.55	15.76	13.31	1.10	1.60	2.61	3.86	7.62	12.63
satimage	6.55	5.02	4.24	0.35	0.51	0.83	1.23	2.43	4.02
letter	9.05	6.94	5.86	0.49	0.71	1.15	1.70	3.35	5.56
motion_sense	4.89	3.75	3.17	0.26	0.38	0.62	0.92	1.81	3.01
mnist	2.87	2.20	1.86	0.15	0.22	0.36	0.54	1.06	1.76

Fig. 3: Ratio of size of each configuration in relation to its baseline.

6 Conclusions

In this work we evaluate the impact of employing Bloom Filters on WNNs to reduce the models' memory footprint on federated learning scenarios. We explored different Bloom Filter variants on nine different classification tasks/datasets and show that the Count-min sketch data structure is one of the best structures for this kind of problem with relatively smaller sizes at reasonable accuracy. Costing at most 3% of accuracy, the Count-min sketch with width of 50 and depth of 2 can be up to 3x smaller than Dict-WiSARD implementations.

References

[1] M Morciniec and R Rohwer. The n-tuple classifier: Too good to ignore. 1995.

[2] T Li, AK Sahu, A Talwalkar, and V Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[3] L Santiago, L Verona, F Rangel, F Firmino, DS Menasché, W Caarls, M Breternitz Jr, S Kundu, P MV Lima, and FMG França. Weightless neural networks as memory segmented bloom filters. *Neurocomputing*, 416:292–304, 2020.

[4] I Aleksander, WV Thomas, and PA Bowden. Wisard· a radical step forward in image recognition. *Sensor review*, 1984.

[5] Massimo De Gregorio. On the reversibility of multidiscriminator systems. Technical report, Technical Report 125/97, Istituto di Cibernetica–CNR, 1997.

[6] Bruno PA Grieco, Priscila MV Lima, Massimo De Gregorio, and Felipe MG França. Producing pattern examples from "mental" images. *Neurocomputing*, 73(7-9):1057–1064, 2010.

[7] IP da Silva, JG dos Santos, and AS Nery. A collaborative weightless neural network. In *2021 Workshop on Communication Networks and Power Systems (WCNPS)*, pages 1–5. IEEE, 2021.

[8] L Fan, P Cao, J Almeida, and AZ Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking*, 8(3):281–293, 2000.

[9] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[10] S Tarkoma, CE Rothenberg, and E Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2011.

[11] A Bacellar, Z Susskind, L Villon, I Miranda, L Santiago, D Dutra, M Breternitz Jr, L John, PMV Lima, and F França. Distributive thermometer: A new unary encoding for weightless neural networks. pages 31–36, 01 2022.