Neurocomputing 555 (2023) 126637

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A conditional branch predictor based on weightless neural networks

Luis A.Q. Villon [a,*], Zachary Susskind [b], Alan T.L. Bacellar [a], Igor D.S. Miranda [c], Leandro S. de Araújo [d], Priscila M.V. Lima [a], Mauricio Breternitz Jr. [e], Lizy K. John [b], Felipe M.G. França [a,f], Diego L.C. Dutra [a]

[a] Universidade Federal do Rio de Janeiro (UFRJ), RJ, Brazil
[b] University of Texas at Austin, TX, USA
[c] Universidade Federal do Recôncavo da Bahia (UFRB), BA, Brazil
[d] Universidade Federal Fluminense (UFF), RJ, Brazil
[e] Instituto Universitário de Lisboa (ISCTE-IUL)/ISTAR, Lisboa, Portugal
[f] Instituto de Telecomunicações, Lisboa, Portugal

ARTICLE INFO

Keywords:
Weightless neural network
WiSARD
Branch prediction
Binary classification

ABSTRACT

Conditional branch prediction allows the speculative fetching and execution of instructions before knowing the direction of conditional statements. As in other areas, machine learning techniques are a promising approach to building branch predictors, e.g., the Perceptron predictor. However, those traditional solutions demand large input sizes, which impose a considerable area overhead. We propose a conditional branch predictor based on the WiSARD (Wilkie, Stoneham, and Aleksander's Recognition Device) weightless neural network model. The WiSARD-based predictor implements one-shot online training designed to address branch prediction as a binary classification problem. We compare the WiSARD-based predictor with two state-of-the-art predictors: TAGE-SC-L (TAgged GEometric-Statistical Corrector-Loop) and the Multiperspective Perceptron. Our experimental evaluation shows that our proposed predictor, with a smaller input size, outperforms the perceptron-based predictor by about 0.09% and achieves similar accuracy to that of TAGE-SC-L. In addition, we perform an experimental sensitivity analysis to find the best predictor for each dataset, and based on these results, we designed new specialized predictors using a particular parameter composition for each dataset. The results show that the specialized WiSARD-based predictor outperforms the state-of-the-art by more than 2.3% in the best case. Furthermore, through the implementation of specialized predictor classifiers, we discovered that utilizing 90% of the specialized predictor for a specific dataset yielded comparable performance to the corresponding specialized predictor.

1. Introduction

Recently, academia and industry [1] have used neural networks to address several problems and challenges related to computer microarchitecture. Specifically, innovative techniques for implementing conditional branch prediction were covered using perceptron [2,3], feedforward neural networks [4], recurrent networks and convolutional networks [5,6]. Conditional branch prediction is an essential technique and a keystone of modern superscalar computer processors. This type of prediction uses a dedicated branch predictor unit implemented in hardware. Those predictors aim to identify patterns in the execution history of a program to predict the outcome of a particular branch in the instruction stream. An increment in the branch predictor accuracy is a relatively simple and effective way to enhance performance and reduce energy consumption [7]. Also, the area and energy costs of the branch predictor unit are key considerations in the microprocessor design.

Weightless neural networks (WNNs) are a category of neural models which use neurons called RAM nodes to perform prediction. The neurons are made up of lookup tables (LUTs) and do not perform complex arithmetic operations. One main advantage of WNNs RAM nodes is the ability to learn non-linear functions of their inputs, which is not possible in a conventional weighted neural network, such as the perceptron. The WiSARD (Wilkie, Stoneham, and Aleksander's Recognition Device) [8] is the first WNN to achieve commercial success and is the neural network model adopted in this paper.

Due to the ability to learn non-linear features indirectly represented by the inputs and its relatively simple arithmetic operations, the WiSARD model is an attractive alternative to traditional neural-based

* Corresponding author.
E-mail address: lvillon@cos.ufrj.br (L.A.Q. Villon).

https://doi.org/10.1016/j.neucom.2023.126637
Received 15 February 2023; Received in revised form 22 May 2023; Accepted 24 July 2023
Available online 28 July 2023

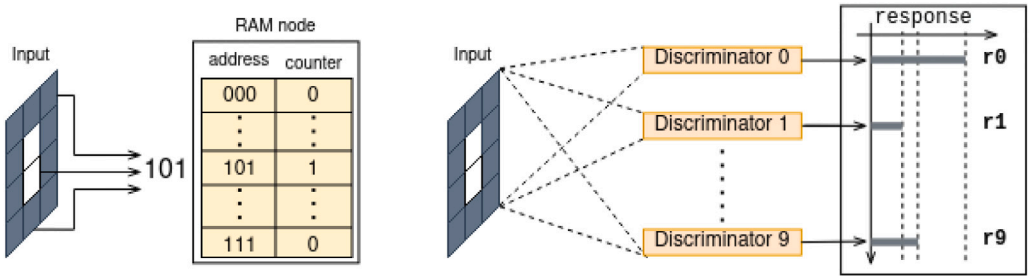


Fig. 1. In this representation of the WiSARD model example, the input image contains "0". It shows an outline of the training phase on the left side. On the right side, the corresponding discriminator produces the strongest response in the classification phase.

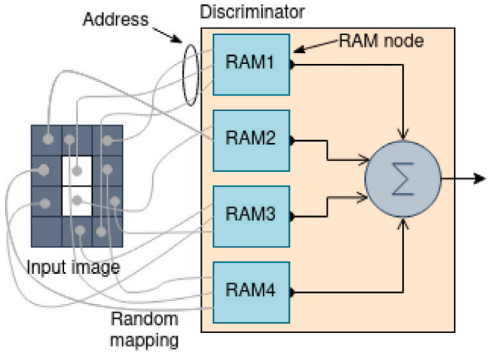


Fig. 2. A representation of the RAM-Discriminator structure.

predictors. Nevertheless, as far as we are aware, there has been no previous work using the WiSARD technique or any weightless neural model to implement a conditional branch predictor. This work aims to explore the potential gain in accuracy and estimated hardware costs of using a WiSARD-based branch predictor, for which we propose a new predictor architecture. We designed this novel predictor to treat branch prediction as a binary classification problem and explore how it performs a one-shot online training methodology.

The rest of this article is organized as follows. We present background and fundamental concepts related to this work in Section 2. Our proposal for the WiSARD-based predictor architecture in Section 3 and in Section 4 the experimental data and the methodology used. Section 5 shows the experimental results and discusses them in detail. Finally, we present conclusions in Section 6.

2. Background

This Section briefly provides the relevant background and fundamental concepts related to conditional branch prediction and the WiSARD model.

2.1. Conditional branch predictor

Conditional branches are employed to make decisions in the execution of a program based on the results of comparison and logical expressions. In high-level programming languages, conditional branches are consequences of conditionals control flow statements, e.g., *if*., *for*, and *while*. The compiler translates those conditional control flow statements to conditional branch instructions. These instructions modify the flow of the program so that the processor is directed to fetch instructions that are not in the sequential order [9]. However, this introduces performance issues for modern speculative and pipelined processor implementations.

To avoid pipeline bubbles and stalls due to conditional branch instructions, nearly all modern high-performance processors implement a conditional branch predictor unit in their microarchitecture design [10]. Instead of stopping execution at a conditional branch until the branch conditions are resolved, a processor uses the branch predictor to fetch and speculatively execute instructions along the predicted path. The main idea consists in predicting branches based on dynamic information provided by the historical behavior of prior executions of a particular conditional branch and related instructions at the microarchitecture level.

In general, a branch predictor has three elements: (1) the input, which conveys information and features about the current instruction, such as branch history for example; (2) the prediction, which communicates to the instruction fetch unit the next instruction it must fetch; (3) and the main predictor architecture, which uses arithmetic or logic operations to compute the prediction.

Branch instructions are frequent, composing nearly 20%–30% of all instructions of computer programs [11]. Furthermore, as computer architectures become more complex with more instructions issued per cycle increases, the penalty for a prediction error (misprediction) increases [2]. Since branch misprediction implies a higher latency and higher energy consumption, a small improvement in branch prediction accuracy can significantly boost performance and energy efficiency [12].

Most modern branch predictors are variants of the TAGE (TAgged GEometric) [13] and/or perceptron branch predictors [2]. In particular, the TAGE-SC-L [14] predictor is considered the state-of-the-art in the industry [15], and it won the 2016 branch predictor championship. In this predictor, the input consists of a large global history register and other microarchitecture features. The history register contains tagged predictor components indexed with distinct history lengths forming a geometric series. TAGE updates the tag after the execution of each branch instruction. In addition, TAGE has a neural-based statistical corrector to detect unlikely predictions and to revert them.

2.2. Neural-based branch predictors

Another state-of-the-art branch predictor is the perceptron-based predictor. The first relevant work used a single-layer perceptron [2] which was later improved in more sophisticated versions [3,16,17]. Research projects to reduce the power consumption, complexity [18–20], and also, to deal with the impossibility, inherent to perceptron models, to learn nonlinear functions from the inputs [21,22] also are found in the literature.

One of the most recent versions is the Multiperspective Perceptron predictor, based on the idea of viewing branch history from multiple

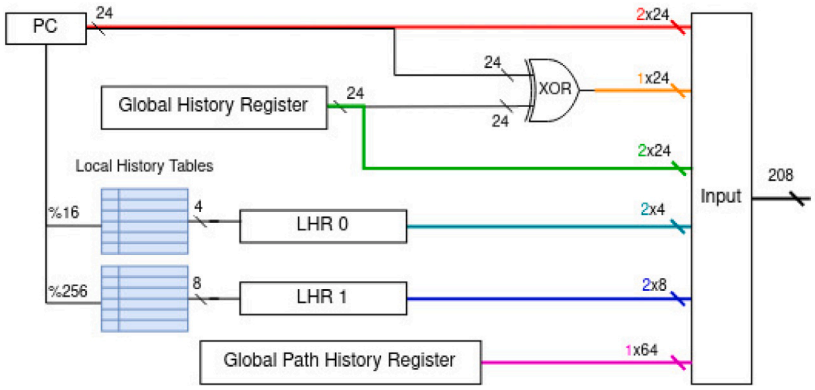


Fig. 3. A depiction of the input composition. In this example the input size is 208 bits.

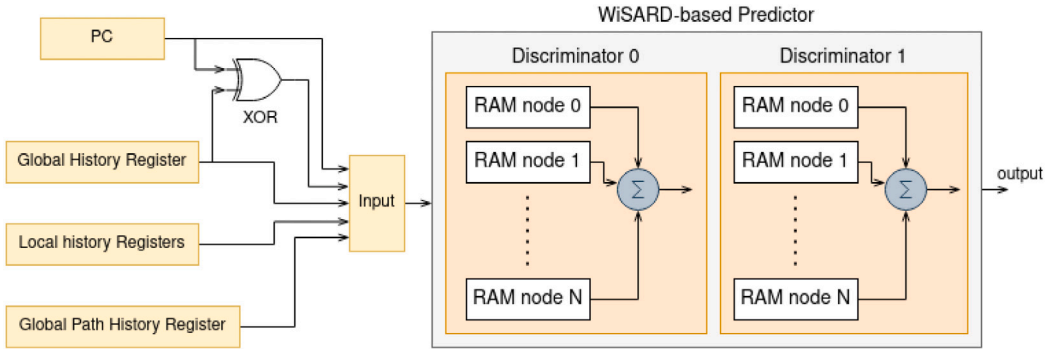


Fig. 4. A depiction of the WiSARD-based predictor. In this example there are three local history registers.

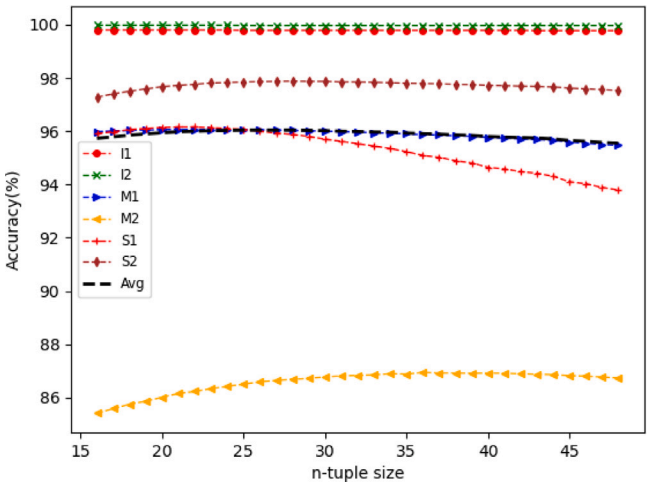


Fig. 5. Results of accuracy obtained by the WiSARD-based predictor in the classification phase as the size of the n-tuple increases. The "Avg" line represents the average accuracy of all datasets. Higher is better.

perspectives [23]. This predictor uses pattern histories and features based on other metrics, which results in large input sizes for the data to be linearly separable. The weights are chosen by hashing across the different features used to make a prediction. The success of perceptron-based predictors confirms that neural networks are effective methods of branch prediction for industrial applications [24].

2.3. Weightless neural networks

Weightless neural networks (WNNs) or *n-tuple classifiers* [25] are a subdivision of machine learning which use neurons based on Random Access Memory (RAM) units to perform prediction. Similarly to traditional artificial neural networks (ANNs) in their early days [26], the human nervous system inspired the WNNs. However, in WNNs, the dendritic tree is prioritized, unlike conventional ANN paradigms that utilize the weighted-sum-and-threshold neurons [27]. This is an important fact since the vast majority of synapses terminate on the neuron's dendritic tree [28]. Although neurons in WNN models do not utilize complex arithmetic operations, these neural models can learn nonlinear functions from the inputs.

2.4. WiSARD

The WiSARD (Wilkie, Stoneham, and Aleksander's Recognition Device) was the first weightless neural network distributed commercially.

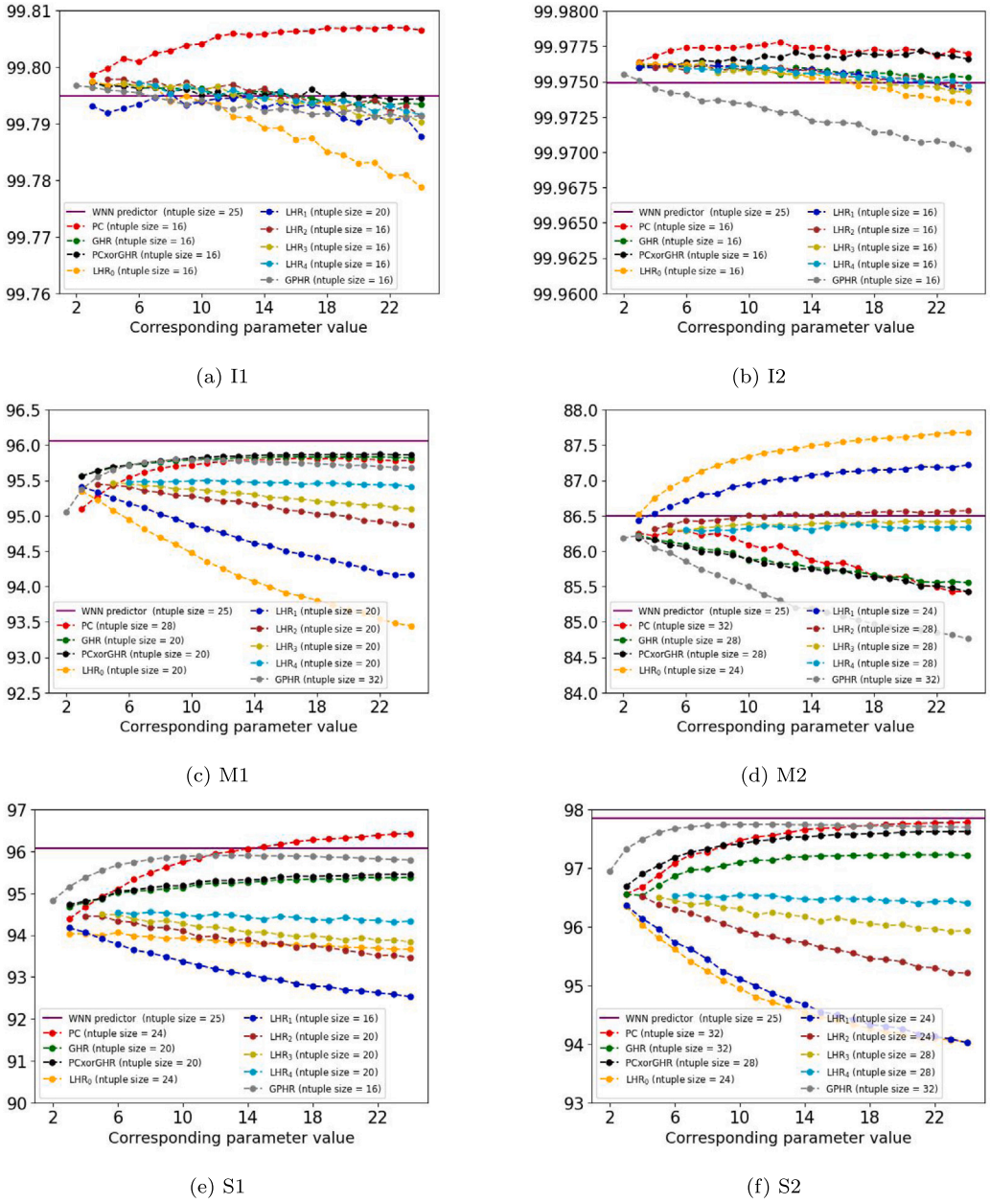


Fig. 6. Sensitivity Analysis for each dataset. The horizontal and vertical axis represent the parameter associated to each feature and the accuracy, respectively. Each curve represents the best result for each feature along Figs. B.9–B.14 from Appendix B.

It consists of a n-tuple classifier composed of class discriminators. Each discriminator is a set of N RAM nodes having n address lines each [29] and trained on a particular class of patterns.

To illustrate how the WiSARD model works, we describe an example implemented for digit recognition tasks, applied to a binarized image in a matrix representation (Fig. 1). The learning phase consists of writing 1's in each RAM node in the respective discriminator that is selected using n address bits randomly (but consistently) extracted from the input pattern value. In the classification phase, all RAM nodes similarly designated by the input, are read. Then the resulting values are added to produce a *response* value. We take the index of the discriminator with the highest response value as the predicted class. To deal with the problem of learning saturation, we implement the contents of the RAM nodes as an access counter that is incremented, during the training phase, with each access. The RAM node counter must have a value greater than a threshold defined by a "bleaching" algorithm [30] that is used to resolve ties during prediction (when discriminator responses are ambiguous because their differences are below a tolerance error). On performing inference, the output of a RAM is "1" if the addressed value is greater than the threshold, otherwise, it is "0". In addition, we present the structure of RAMs in a Discriminator in Fig. 2.

3. Proposal

In this section, we propose a novel predictor architecture based on the WiSARD model, which is adaptable to fit different constraints of the

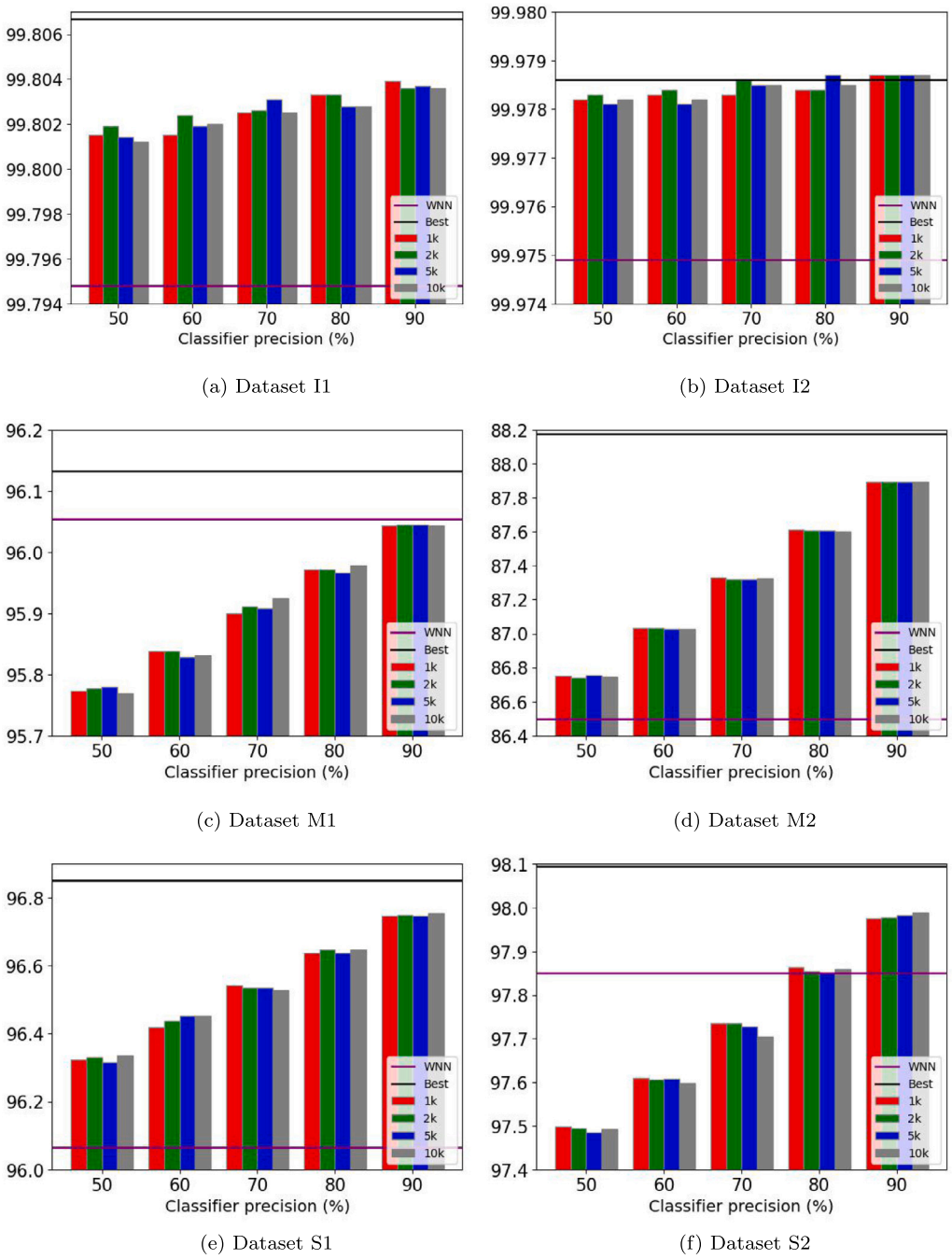


Fig. 7. Behavior of the classifier of the best predictors for each dataset represented in each subfigure, where the X-axis represents the precision of the customized predictor classifier. The accuracy is displayed on the Y-axis in each case. Each bar represents the execution with a certain instruction block size, labeled as 1k, 2k, 5k and 10k, where 'k' means 1000 instructions.

applications. We designed the WiSARD-based predictor to perform one-shot online training, and the respective classification phase performs a binary classification.

3.1. Input composition

The binary input is a linear combination of different microarchitectural information about current and recent branch addresses. The input is expressed as:

$$input = a \cdot PC + b \cdot GHR + c \cdot PCxorGHR + \sum_{i=0}^{N-1} d_i \cdot LHR_i + e \cdot GPHR \quad (1)$$

Where: *PC* (program counter) represents the least significant bits from the memory address of the current branch instruction, *GHR* is the global history register from the last conditional branches outcomes, *PCxorGHR* is the logical exclusive-or operation between the PC and GHR, LHR_i is each of the N local history registers from the local branch

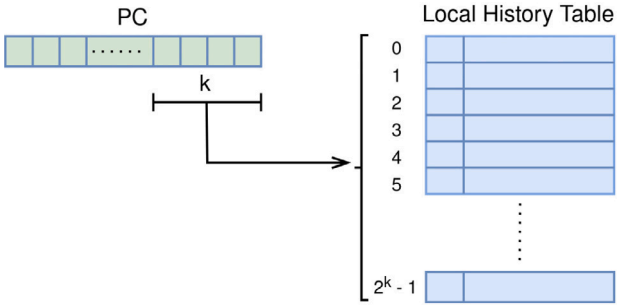


Fig. A.8. Structure and composition of a LHT. Each row in the table represent a particular LHR.

and *GPHR* is the global path history register which stores the eight less significant bits from the last eight conditional branches. We describe a more complete and detailed description of these features in Appendix A. The additional parameters *a, b, c,* d_i and *e* represent the strength of its associated feature in the input.

As an example, suppose we express the values of the variables and parameters as follows: PC = 24, GHR = 24, PCxorGHR = 24, LHR_0 = 4, LHR_1 = 8, GPHR = 64, a = 2, b = 2, c = 1, d_0= 2, d_1 = 2, e = 1. From an architectural point of view and a hardware perspective, Fig. 3 shows how we use the parameters and registers to compose the input. In this example, the input size is 208 bits.

In System Verilog language, we can represent the input by the following relation using replication concatenation and exclusive-or operators.

$$input = \{\{2\{PC[23:0]\}\},\{1\{PC[23:0] \wedge GHR\}\},\{2\{GHR\}\},\{2\{LHR0\}\},\{2\{LHR1\}\},\{1\{GPHR\}\}\} \quad (2)$$

3.2. Predictor architecture

We show the WiSARD-based predictor architecture in Fig. 4. At the beginning, the RAM node counters are initialized with zero contents. The classification phase occurs first since the predictor uses an online learning methodology. In this phase, we pseudo-randomly divided the current input information in *n-tuples of bits* to get the address of a RAM node located in two discriminators: Discriminator "0", which represents a not taken branch and Discriminator "1" otherwise. We generate a response from both discriminators, and the one with the highest response value determines the corresponding final output. In addition, we implemented a bleaching algorithm, which sets a threshold that must be exceeded every time there is a tie in the classification process.

Once the classification phase for the current input finishes, next comes the training phase, where the input is again split in *n-tuples* of bits to get the address of all RAM nodes located in the respective Discriminator. Then the counters in each designated RAM node are updated accordingly. This procedure, including the classification and training phase, is performed for all the subsequent inputs of a given dataset.

4. Methodology

4.1. Dataset

As dataset for this study, we use the 3rd Championship Branch Prediction (CBP-3) organized by the JILP Workshop on Computer Architecture Competitions (JWAC) from Kaggle [31]. The information is composed only of conditional branch information, and it is distributed in 3 categories, according to the benchmark application class: integer workloads (I1 and I2), multimedia (M1 and M2), and server (S1 and S2) applications. All of them have 4×10^5 conditional branch instructions, with the exception of dataset M1, which has 3×10^5 elements.

Among these datasets, we only consider the PC and the actual outcome of each branch. This is used to build the microarchitecture information that compounds the input, as described in Section 3 and Appendix A. Moreover, as we design our WiSARD-based branch predictor assuming already existing hardware structures of the branch predictor unit, i.e., Local History Table and Global History Register, we use the whole available dataset to evaluate our proposed designs and the existing solutions, as they all assume the branch unit works using online learning.

4.2. Experimental setup

We performed 100 experiments on each group of datasets. The quantitative results and plots, shown later, represent the average of 100 values. We choose this number to evaluate how the uniform distribution of inputs over RAMs impacts the overall performance of our design. Even though a given final hardware implementation must have a fixed mapping, this approach allows us to understand how our results depend on the input mapping. We set a fixed size, in bits, for the features in the input from Eq. (1), as follows: PC = 24, GHR = 24, PCxorGHR = 24, LHR_0 = 24, LHR_1 = 16, LHR_2 = 9, LHR_3 = 7, LHR_5 = 5, GPHR = 64. Therefore, we express the input in a more simplified way:

$$input = 24a + 24b + 24c + 24d_0 + 16d_1 + 9d_2 + 7d_3 + 5d_4 + 64e \quad (3)$$

In the rest of this work, we will use Eq. (3) for all different experimental scenarios. Furthermore, we compare our solution against the TAGE-SC-L and the Multiperspective Perceptron predictors on all datasets. The input size for both predictors is 3127 and 2329 bits, respectively, and their training and classification phase do not use a random process, as they are final hardwired architecture implementation models.

5. Results and discussion

We show in this section the results from four different but complementary experimental approaches. In the first part, we made a pseudo-exhaustive hyperparameter search to find the best input composition for our predictor whose accuracy would outperform, on average, the state-of-the-art predictors. We must point out that the current state-of-the-art branch predictors area achieve performances in the high 99%s for some relevant benchmarks. Moreover, because of the sheer amount of instructions executed in a CPU quantum or time slice, that can have as much as $25,000,000$ branches in a 1 GHz microprocessor and, due to the complexities of superscalar processors, any fluctuation in the branch predictor accuracy causes a relevant impact in the overall system performance. Subsequently, we performed a sensitivity analysis of all features that compound the input to explore the particular behavior and trends. Then, we obtain the parameter configuration to find the best potential and specialized predictor for each dataset. In the next step, we performed an experimental analysis of specialized predictor classifiers.

5.1. Best result - preliminary exploration

Preliminary, our pseudo-exhaustive search (given that WNNs allows for very agile implementations) showed that the best configuration for the parameters is: $a = 24, b = 12, c = 12, d_0 = 8, d_1 = 8, d_2 = 8, d_3 = 6, d_4 = 12, e = 8$. Thus, according to Eq. (3), the size of this input is: 24 ·24 + 12 ·24 + 12 ·24 + 8 ·24 + 8 ·16 + 8 ·9 + 6 ·7 + 12 ·5 + 8 ·64 = 2158 bits, which is smaller than the TAGE-SC-L and the Multiperspective Perceptron counterparts.

We present the details of this first experimental result in Fig. 5. It illustrates how the accuracy varies as the size of the *n*-tuple increases.

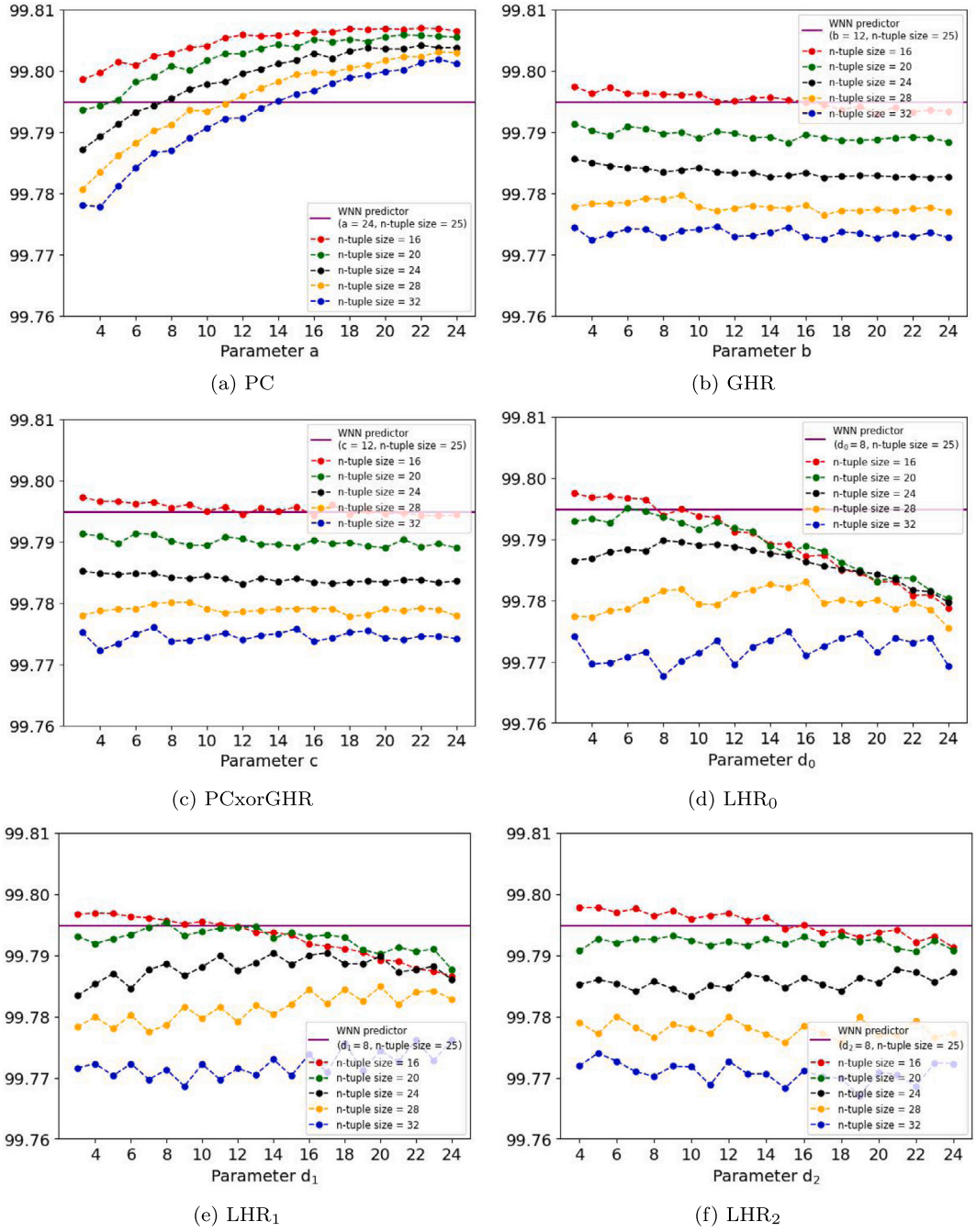
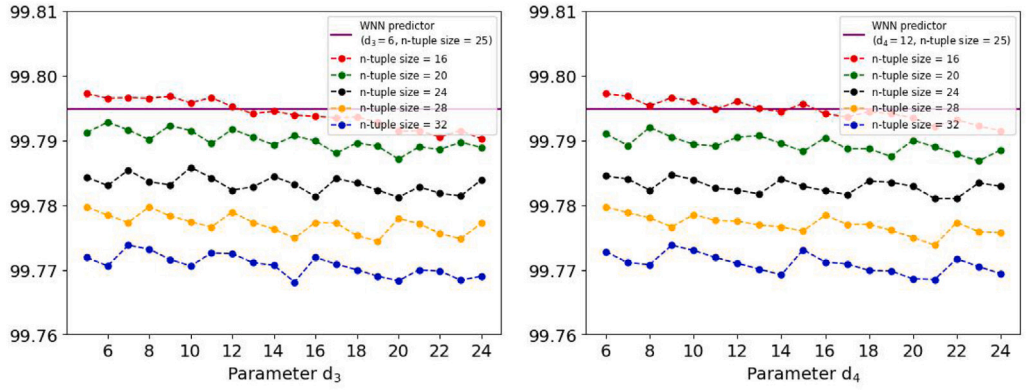


Fig. B.9. Sensitivity Analysis for dataset I1. We observe that PC is the most important feature (Fig. B.9a) since the accuracy increases as its corresponding parameter a increases when compared to the other features (Figs. B.9b–B.9i). For the other features, the accuracy drops or remains oscillating (Figs. B.9b–B.9h). The exception is the feature GPHR, in which the initial behavior depends on the n-tuple sizes and for large values of e the accuracy remains nearly constant (Fig. B.9i). In addition, in nearly all cases, the accuracy drops as n-tuple size increases.

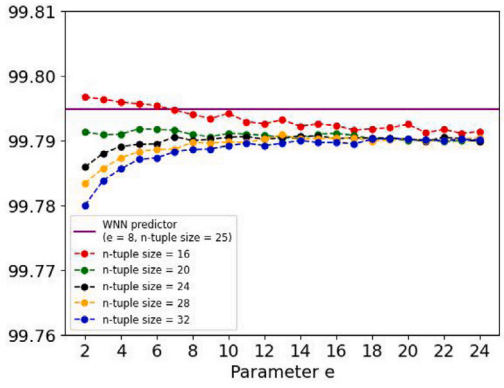
First, we notice that the accuracy in the datasets I1 and I2 remains almost constant. In datasets M1 and S1, the accuracy increases up to n-tuple size = 22 and then decreases, being dataset S1 where we observe this effect more pronounced. On the other hand, in datasets S2 and M2, we see a more prominent accuracy benefit. On average (black line), the accuracy increases up to n-tuple size = 25.

We compared these results with TAGE-SC-L and Multiperspective Perceptron (shown in Table 1 where WNN, T, and MP stand for the WiSARD-based, TAGE-SC-L, and Multiperspective Perceptron predictors respectively). We extended the results to a precision of four decimal places to illustrate a more complete exploration. On average, the WiSARD-based predictor achieves approximately the same accuracy as the TAGE-SC-L and slightly outperforms the Multiperspective Perceptron by approximately 0.09%. We emphasize that our predictor shows a higher accuracy value on the dataset M2 compared to all other predictors.

In addition, we performed supplementary experiments using the same input from TAGE-SC-L and Multiperspective Perceptron in the WiSARD-based predictor. The results are shown in Table 2. Interestingly, the accuracies obtained with these inputs, on average, were 77.1948% and 89.7698% respectively. As expected, our predictor has



(g) LHR_3 (h) LHR_4



(i) GPHR

Fig. B.9. (*continued*).

Table 1
Accuracy results compare the best configurations of the WiSARD-based predictor with the state-of-the-art (TAGE-SC-L) and the Multiperspective Perceptron predictor. We labeled them WNN, T, and MP, respectively.

Predictor	I1	I2	M1	M2	S1	S2	Average
WNN	99.7948 ± .0016	99.9749 ± .0010	96.0540 ± .0224	86.4968 ± .1458	96.0651 ± .0349	97.8504 ± .0124	96.0393 ± .0621
T	99.8138 ± .0000	99.9782 ± .0000	96.1357 ± .0000	85.8582 ± .0000	96.3213 ± .0000	97.8710 ± .0000	95.9964 ± .0000
MP	99.7700 ± .0000	99.9792 ± .0000	96.2340 ± .0000	85.7533 ± .0000	96.2143 ± .0000	97.7645 ± .0000	95.9525 ± .0000

Table 2
Accuracy results of the WiSARD-based predictor by using the inputs from the TAGE-SC-L and the Multiperspective Perceptron predictors, labeled as Input-T and Input-MP respectively. For the sake of comparison, the bottom row, labeled as Input-W, represents the best input configurations of the WiSARD-based predictor from Table 1.

Input	I1	I2	M1	M2	S1	S2	Average
Input-T	96.2828 ± .1219	99.8607 ± .0011	62.8316 ± .2016	69.4551 ± .0404	64.5249 ± .0415	82.2137 ± .0540	79.1948 ± .0103
Input-MP	99.6334 ± .0220	99.9342 ± .0012	82.8533 ± .2203	74.3839 ± .4133	87.1223 ± .0692	94.6945 ± .0188	89.7698 ± .0375
Input-W	99.7948 ± .0016	99.9749 ± .0010	96.0540 ± .0224	86.4968 ± .1458	96.0651 ± .0349	97.8504 ± .0124	96.0393 ± .0621

a completely different knowledge acquisition process than the other predictors since the discrepancy in accuracy is considerable.

5.2. Sensitivity analysis

We performed a sensitivity analysis in order to determine the most relevant features that comprise the input. We carried out this study across all the six datasets. In all scenarios, the parameters for the base case were: $a = 2, b = 2, c = 2, d_0 = 2, d_1 = 2, d_2 = 3, d_3 = 4, d_4 = 5, e = 1$.

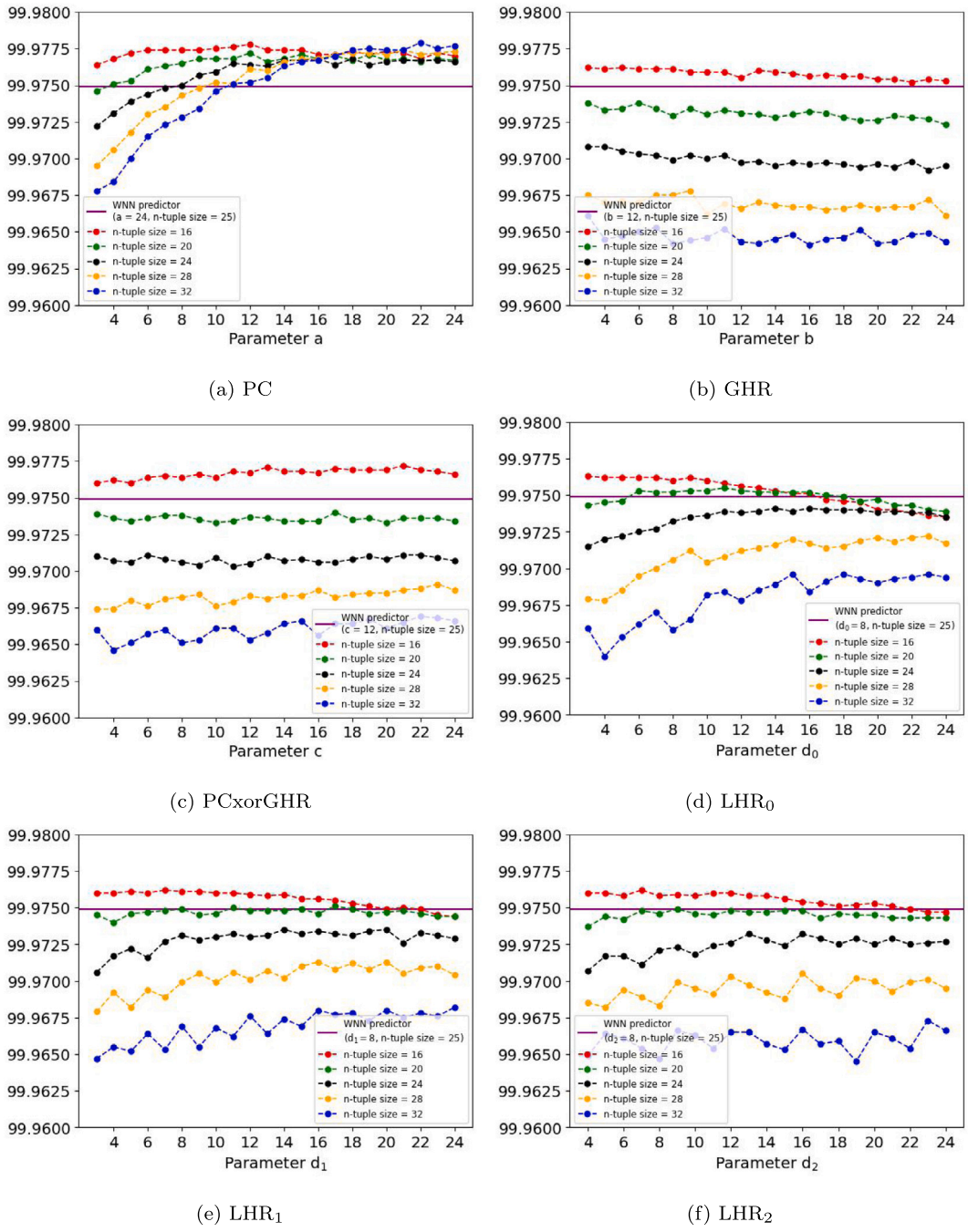
We show and explain the full results in Appendix B. In Fig. 6, we report the best curves achieving the highest accuracy for each feature and dataset. In the legend, the n-tuple size utilized in each curve is indicated within parentheses for each case.

In datasets I1 and I2, we observe that PC is the most relevant feature (Figs. 6(a) and 6(b)) since the accuracy increases as its corresponding parameter also increases. On the other hand, for the other features, the accuracy drops smoothly. The exceptions are features LHR_0 and GPHR, in which the accuracy decreases quickly in I1 and I2, respectively.

We observe a different situation in datasets M1 and M2. In both datasets, the accuracy increases for features PCxorGHR and LHR_0 respectively (Figs. 6(c) and 6(d)). Interestingly in M1, the worst trends are characterized by all LHRs, specifically LHR_0. While in M2, the accuracy drops significantly for the feature GPHR.

In addition, in datasets S1 and S2, the most relevant features are GPHR and PC for large value of their corresponding parameter (Figs. 6(e) and 6(f)). The accuracy also increases for features GHR and PCxorGHR while it decreases significantly for the features LHR_1 and LHR_0 in datasets S1 and S2, respectively.

These results show that datasets that correspond to the same category have the same correlation. We present this for the group integers



(a) PC (b) GHR

(c) PCxorGHR (d) LHR_0

(e) LHR_1 (f) LHR_2

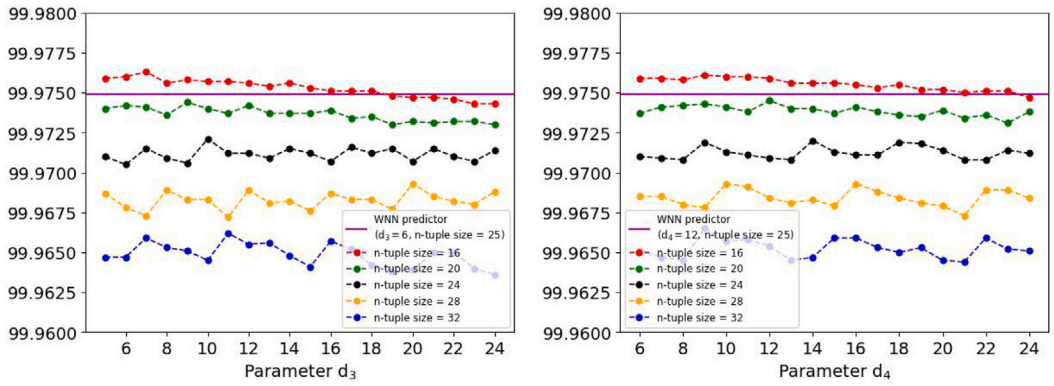
Fig. B.10. Sensitivity Analysis for dataset I2. In this case the PC and the PCxorGHR are the most relevant features since the accuracy increases as its corresponding parameter a and c increases when compared to the other features (Figs. B.10a and B.10c). In nearly all cases (Figs. B.10b–B.10i) the accuracy decreases as the n-tuple size increases. For LHR_2, LHR_3, LHR_4 features, the accuracy keeps fluctuating as the corresponding parameter increases in the largest n-tuple sizes (Figs. B.10f–B.10h).

(datasets I1 and I2) and server (datasets S1 and S2). Nevertheless, this behavior went unobserved in the multimedia category (datasets M1 and M2).

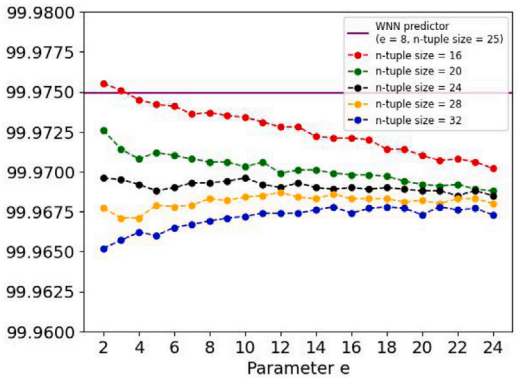
Among all these results, we must highlight the cases in which this analysis exceeds the precision previously obtained in our first experiment (Table 1). Clearly, in datasets I1 and I2, there is at least one curve that surpasses our previous result for some value of the parameters (Figs. 6(a) and 6(b)). In datasets M1 and S2, there is no such a curve that outperforms the horizontal line (Figs. 6(c) and 6(f)). Lastly, in dataset M2, the results can be significantly more than 1% better than the previous WNN-predictor result (Fig. 6(d)); while in dataset S1, this occurs for large values of the parameter associated with PC (Fig. 6(e)).

5.3. Best predictor for each dataset

Based on these previous results, we perform a second pseudo-exhaustive hyperparameters search to find the best particular results for each dataset. We present the results obtained in this study in Table 3, and we consolidate them with the results of Table 1 in Table 4. In the experimental approach of this section, it is more important to outperform the accuracy described in Table 1 versus the necessity of smaller input sizes. We achieve this objective for all datasets compared to the previous WNN version, while our predictors only outperform the state-of-the-art in the datasets M2, S1, and S2. We emphasize the results in dataset M2, where the accuracy obtained is at least 2.3% higher than TAGE-SC-L and Multiperspective Perceptron counterparts.



(g) LHR_3 (h) LHR_4



(i) GPHR

Fig. B.10. (*continued*).

Table 3
Accuracy results for the best parameters configuration for each dataset.

Dataset	n-tuple	a	b	c	d_0	d_1	d_2	d_3	d_4	e	Input size	Accuracy
I1	17	24	2	2	2	8	3	4	5	1	992	99.8067 ± .0018
I2	27	24	6	6	2	2	3	4	0	2	1127	99.9786 ± .0009
M1	24	72	24	72	16	16	16	12	24	16	6044	96.1322 ± .0157
M2	23	10	10	10	150	10	15	20	25	5	5200	88.1783 ± .0202
S1	38	140	10	10	4	4	4	10	12	8	4678	96.8521 ± .0179
S2	38	150	16	16	2	2	8	16	26	8	5274	98.0946 ± .0110

Table 4
Comparison of the accuracy results reported in Tables 1 and 3.

Dataset	Accuracy(Best)	Accuracy(WNN)	Accuracy(T)	Accuracy(M)
I1	99.8067 ± .0018	99.7948 ± .0016	99.8138 ± .0000	99.7700 ± .0000
I2	99.9786 ± .0009	99.9749 ± .0010	99.9782 ± .0000	99.9792 ± .0000
M1	96.1322 ± .0157	96.0540 ± .0224	96.1357 ± .0000	96.2340 ± .0000
M2	88.1783 ± .0202	86.4968 ± .1458	85.8582 ± .0000	85.7533 ± .0000
S1	96.8521 ± .0179	96.0651 ± .0349	96.3213 ± .0000	96.2143 ± .0000
S2	98.0946 ± .0110	97.8504 ± .0124	97.8710 ± .0000	97.7645 ± .0000

When comparing the results from Table 3 to the sensitivity analysis (Figs. B.9–B.14), a direct correlation between the parameters and the features from the input it is observed, as expected. We summarize this correlation in Table 5. By far, the PC is the most relevant feature.

5.4. Analysis of specialized predictor classifiers

Since the WiSARD-based predictor outperforms the other state-of-the-art predictors significantly in some cases, it opens several avenues to research the behavior of a classifier of specialized predictors for each dataset corresponding to particular applications, as proposed in a

Table 5
Most relevant feature of the input for each dataset.

Dataset	Most important feature(s)	Least important feature(s)
I1	PC	LHR_0
I2	PC	GPHR
M1	PC, PCxorGHR	LHR_0
M2	LHR_0	GPHR
S1	PC, GPHR	LHR_1
S2	PC, GPHR	LHR_0, LHR_1

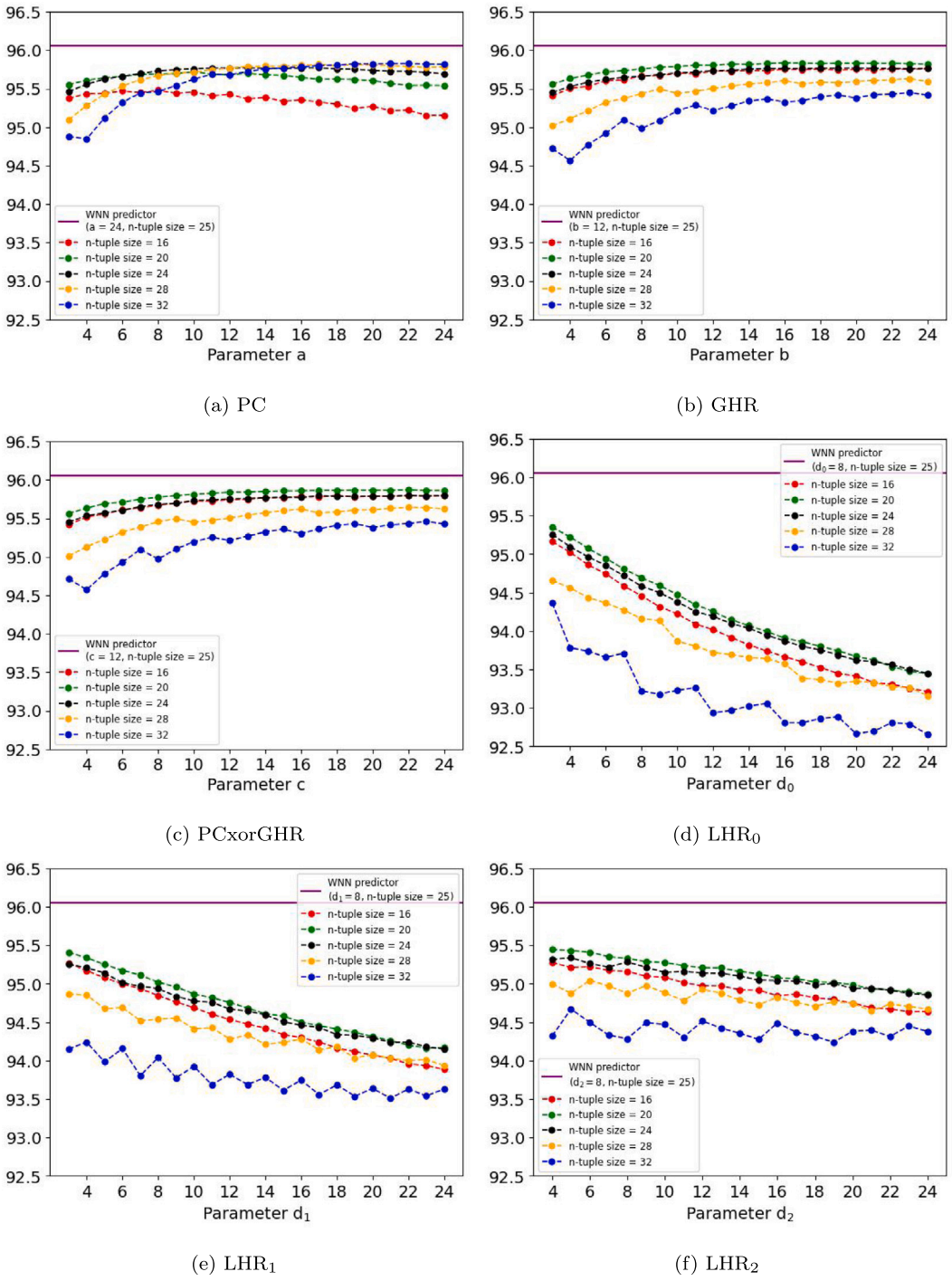
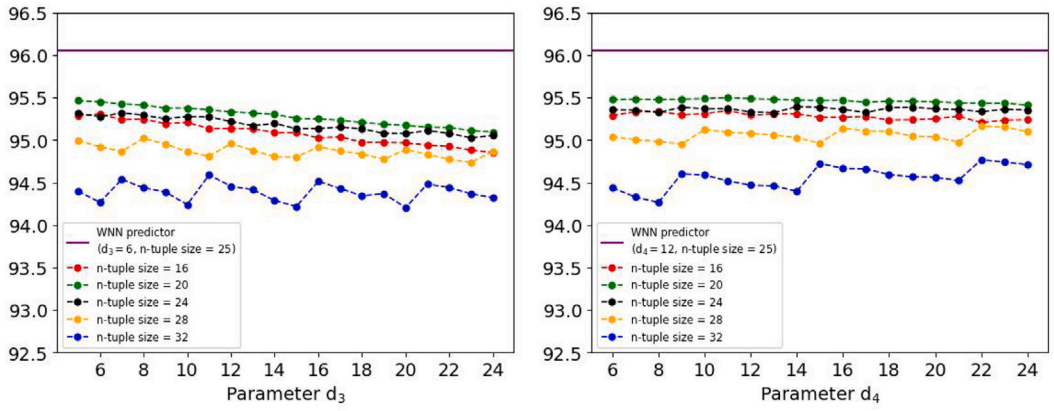


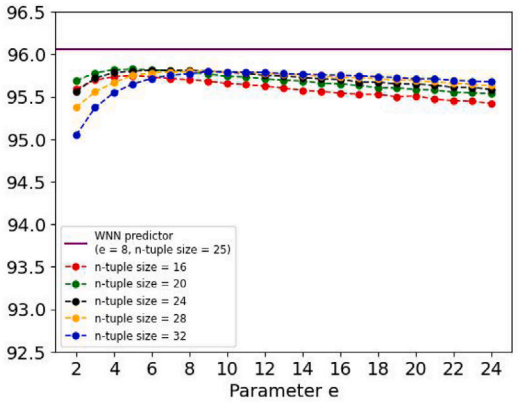
Fig. B.11. Sensitivity Analysis for dataset M1. The results show differences in accuracy trends for each feature in each case. The accuracy increases or decreases smoothly for the features PC, GHR and PCxorGHR (Figs. B.11a–B.11c), but substantially drops for LHR_0, LHR_1, LHR_2 (Figs. B.11d–B.11f). Interestingly, the accuracy initially increases and then drops smoothly for large values of e for GPHR (Fig. B.11i). In most cases, the accuracy decreases for large n-tuple sizes (Figs. B.11b–B.11h).

related work [32]. Thus, we performed an additional analysis using the best predictors for each dataset, according to the results from Table 3.

The main goal is to identify the potential gain in accuracy when all the six category-specific specialized predictors are implemented and used to perform prediction for each corresponding dataset. We envisioned a computer system having a classifier that can, with a given probability, select the correct specialized predictor for a given application. From a computer architecture perspective, the Operating System



(g) LHR_3 (h) LHR_4



(i) GPHR

Fig. B.11. (*continued*).

(OS) can inform the processor core of the optimal predictor during the process scheduling or even be a specialized unit inside the processor core doing the classification according to the current behavior of the process. The classifier in our setup can choose among the six existing datasets in the three categories the one that best matches the behavior of a given program.

We implemented the classification algorithm of specialized predictors by defining the rate of precision of its selection. To quantify the final accuracy of the prediction, we divide the number of instructions in each dataset into blocks of a fixed size, where the prediction of each block of instructions where performed by all the specialized predictors. We summarize these results in Fig. 7. The results of the specialized predictors and the initial version of the WiSARD-based predictor (Table 1), are also displayed by means of horizontal solid lines and are labeled as “Best” and “WNN” respectively.

In datasets I1 and I2, the use of at least 50% of the specialized predictor outperforms the WNN predictor up to more than 0.006% and 0.003%, respectively, (Figs. 7(a) and 7(a)). Nevertheless, in dataset I2, the prediction accuracy with a 90% precision classifier ties with the corresponding specialized predictor in this case.

Moreover, in dataset M1, the use of the classifier did not improve the necessary accuracy to outperform the WNN predictor version. Meanwhile, in dataset M2 the use of 50% and 90%, the specialized predictor surpasses the WNN version by more than 0.2% and 1.4%, respectively. In both datasets, the use of the classifier fails to approach the accuracy of the best results obtained by the corresponding specialized predictor.

In addition, in dataset S1, the use of 90% of the specialized predictor outperforms the WNN predictor up to more than 0.7%. In dataset S2, with the prediction accuracy of the classifier at 80% and 90% of precision, respectively, the presented results tie and outperform by more than 0.1% the accuracy of the corresponding specialized predictor.

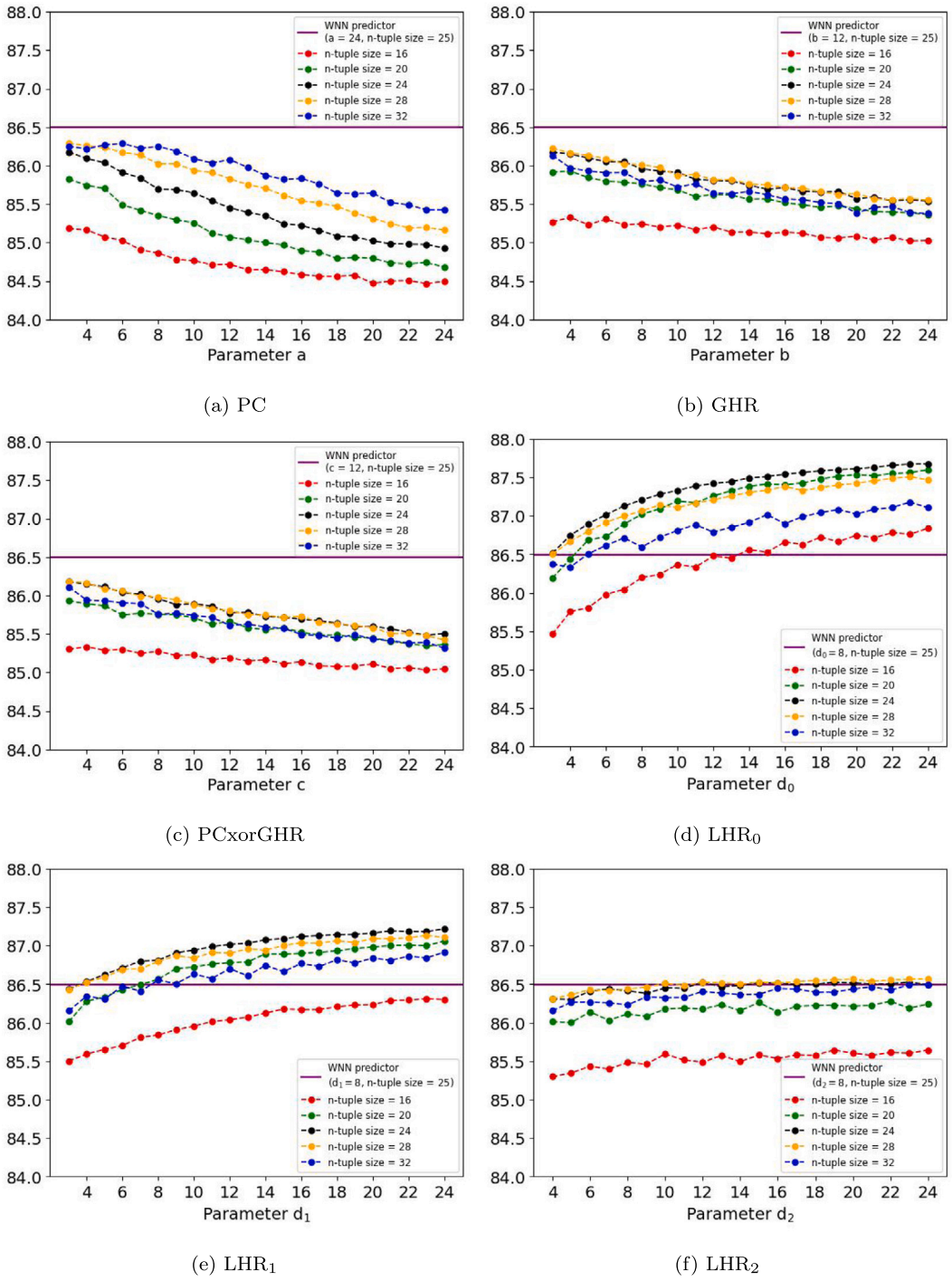
Finally, we also observed that the presented results are independent of the size of the blocks of instructions, as the accuracy remains almost similar in each group of the bar charts among all datasets (Figs. 7(a)–7(f)).

6. Conclusion

WiSARD is one of the most important WNN models, which are neural networks based on RAM that do not perform complex arithmetic operations, and as a consequence, we can implement it in hardware and real-time applications. One interesting and potential area of application of WNN is computer architecture. Specifically, we can explore WNN as part of the conditional branch predictor architecture, a well-established technology implemented in nearly all modern computer processors.

In this work, we proposed and evaluated a conditional branch predictor based on WNNs, particularly in the WiSARD model. We performed four different experiments to obtain a complete exploration of general potential, plus some particular insights.

First, through a pseudo-exhaustive hyperparameter search, we experimented with the WiSARD-based predictor to compare it with TAGE-SC-L, a state-of-the-art, and with the Multiperspective Perceptron, a neural-based predictor. Using a smaller input size (and thus taking



(a) PC (b) GHR

(c) PCxorGHR (d) LHR_0

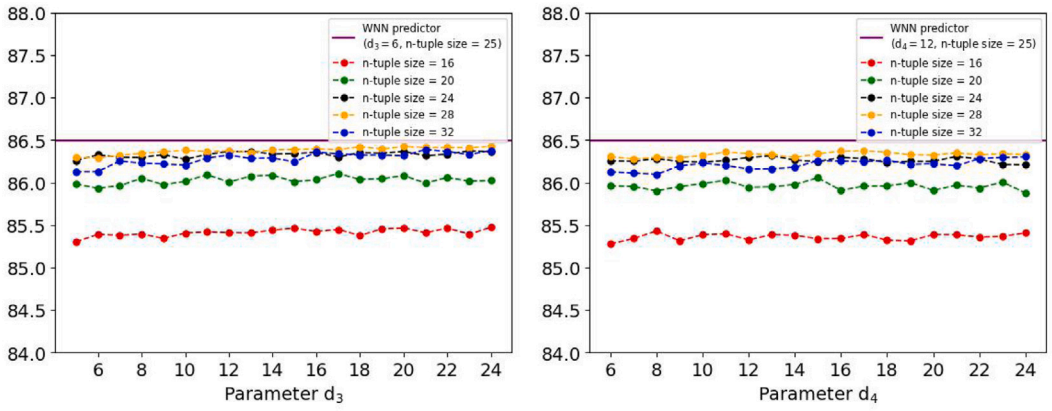
(e) LHR_1 (f) LHR_2

Fig. B.12. Sensitivity Analysis for dataset M2. In this case, the accuracy increases for LHR_0, LHR_1, LHR_2 (Figs. B.12d–B.12f); remains nearly constant for the features LHR_3 and LHR_4 (Figs. B.12g and B.12h) and drops significantly for the other features (Figs. B.12a, B.12b, B.12c, B.12i). For this dataset it is important to highlight the feature GPHR, since the accuracy degrades significantly as the parameter e increases (Fig. B.12i). In addition, in almost all cases the accuracy increases as the n-tuple size increases.

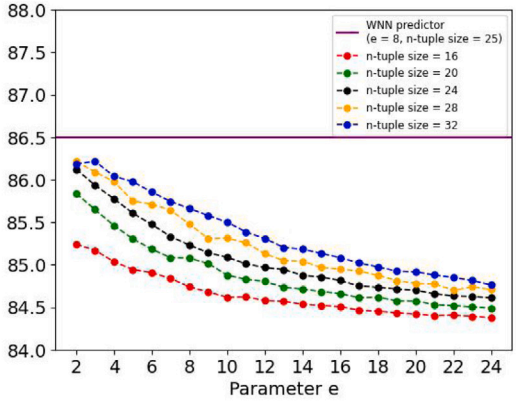
fewer hardware resources), our predictor achieves, on average, similar accuracies to the TAGE-SC-L and outperforms the Multiperspective Perceptron by approximately 0.09%.

Next, we performed a sensitivity analysis in all datasets to determine the most relevant features of the input. The results show that the PC value is the most important feature at the microarchitecture level to our predictor.

Subsequently, our third experimental results show that a deeper pseudo-exhaustive parameter search for each dataset leads to different configurations for our first WiSARD-based predictor, outperforming the



(g) LHR_3 (h) LHR_4



(i) GPHR

Fig. B.12. (*continued*).

TAGE-SC-L and the Multiperspective Perceptron for three datasets. The difference in accuracy for the best case is higher than %2.3.

In addition, since the use of predictor configurations adapted to specific dataset characteristics indicated a promising new venue for further performance gains, we designed specialized predictor classifiers, that with a certain probability, select the correct specialized predictor for an application. Our experiments demonstrated that employing specialized predictors in at least 50% of the branches in our datasets yielded superior results compared to our initial WiSARD-based predictor across four of the six datasets analyzed. Notably, in one specific case, utilizing the specialized predictor on 90% of the branches achieved comparable performance to its corresponding specialized predictor.

We can extend this work by using Bloom filters [33] to reduce the hardware area of our design while reducing memory and power consumption, making the training and classification phases more efficient. A crucial aspect for expanding this study involves investigating various feature selection methods to compare the sensitivity analysis using novel experimental approaches. As the predictor utilizes an online training approach that relies on binary data, which we interpreted as categorical information, the feature selection techniques applicable to this research may diverge from the sensitivity analysis. This comprehensive examination of the approach will be the subject of future research.

Finally, based on the results obtained for the branch predictor problem, we believe the WiSARD model is a good fit for other types of predictors used in computer architecture, specifically at the microarchitecture layer. Since our work shows that WNN can be, at least, explored in this area, we surmise that we are at the start of an interesting research field, and further research is warranted.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank CAPES, Brazil and CNPq, Brazil for the financial support to this work. In addition, this work was partially supported by Fundação para a Ciência e a Tecnologia, I.P. (FCT), Portugal [ISTAR Projects: UIDB/04466/2020 and UIDP/04466/2020], by project FLOYD:POCI-01-0247- FEDER-045912, and Project FCT UIDBASE/50008/2020.

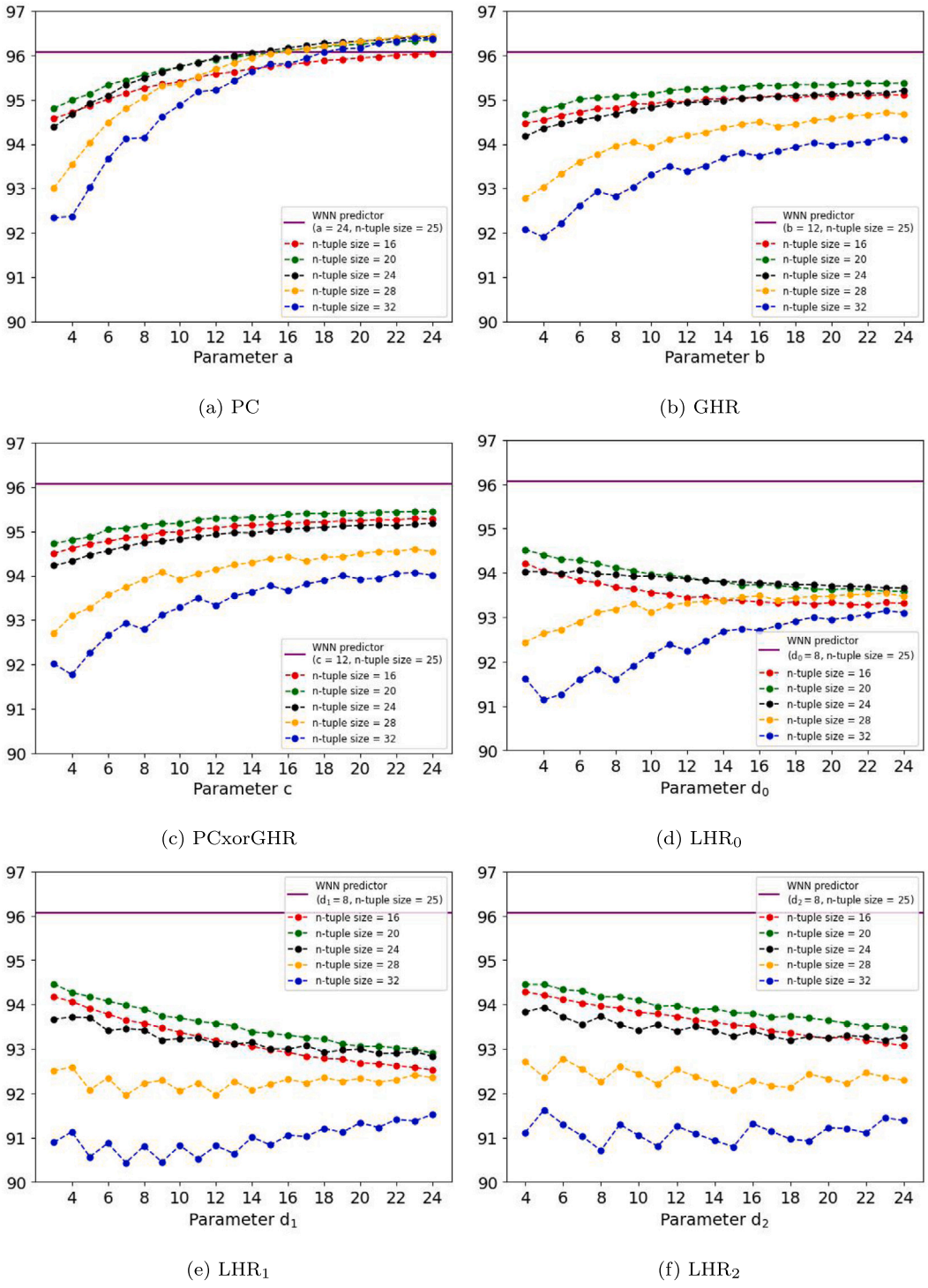


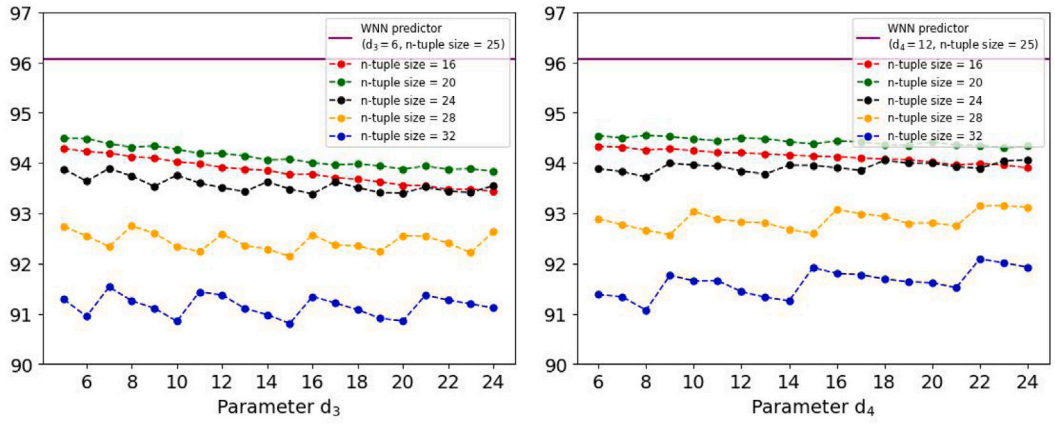
Fig. B.13. Sensitivity Analysis for dataset S1. We observe that the most relevant features are PC and GPHR (Figs. B.13a and B.13i) since accuracy achieves the highest values, particularly for high values of the parameter *a*. The accuracy also increases for the features GHR and PCxorGHR (Figs. B.13b and Fig. B.13c) but decreases for all other LHR type features (Figs. B.13d–B.13h). Furthermore, in almost all cases the best and worst curves correspond to n-tuple sizes 20 and 32 respectively.

Appendix A. Description of features

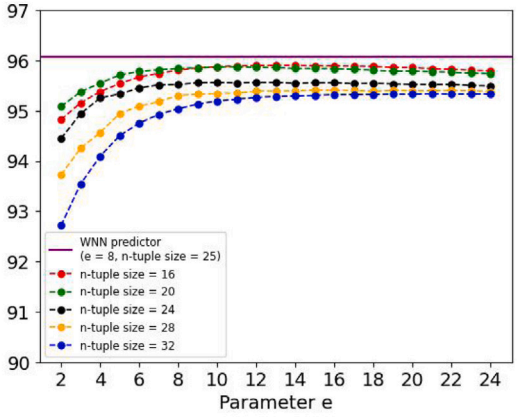
In this appendix, the features that compound the input are described using a computer architecture approach.

A.1. Program counter (PC)

The PC is a register which contains the address bits of the current instruction being executed in a given program. In most modern general



(g) LHR_3 (h) LHR_4



(i) GPHR

Fig. B.13. (*continued*).

purpose processors, the size of the PC is 32 or 64 bits. In the datasets analyzed in this work, the size is 32 bits.

A.2. Global history register (GHR)

The idea of GHR is to track the global history of all conditional branches real outcomes, which can be 0 (Not taken branch) or 1 (Taken branch). This information is stored in a shift register which is updated with the result of the actual branch outcome in the execution of a program.

A.3. XOR operation between PC and GHR

As mentioned and previously addressed in related works [34], the XOR operation (exclusive-OR) between PC and GHR can synthesize the information in a smaller memory space. This leads to a more compact branch predictor unit.

A.4. Local history register (LHR)

This register stores the last occurrences of the same branch instruction. One LHR is associated with one or a particular set of conditional branches. The Local History Registers for all branches are contained in a Local History Table (LHT), in which each entry is indexed by the branch instruction address [35]. Fig. A.8 illustrates how the least significant bits of the PC are employed to update the LHR in the LHT.

A.5. Global path history register (GPHR)

This memory represents an array of the last 8 branch addresses. As branches are executed, their addresses are shifted into the first position of this array. In this work, the elements of the array are simply the lower 8 bits of the branch address. In other related works, this register is known as Global Addresses (GA) [15,36].

Appendix B. Supplementary sensitivity analysis

In this appendix we show the full results of the sensitivity analysis exhibited in Figs. B.9–B.14. In all subfigures, the vertical and horizontal axis represent, respectively, the accuracy and the parameter associated to each feature according to relation (1). Each dashed curve represent the behavior for a particular n-tuple size. In addition, the continuous horizontal line exhibits the previous experimental result (Table 1). For each feature, this horizontal line also references, in parentheses, the corresponding parameter value and the n-tuple size previously employed.

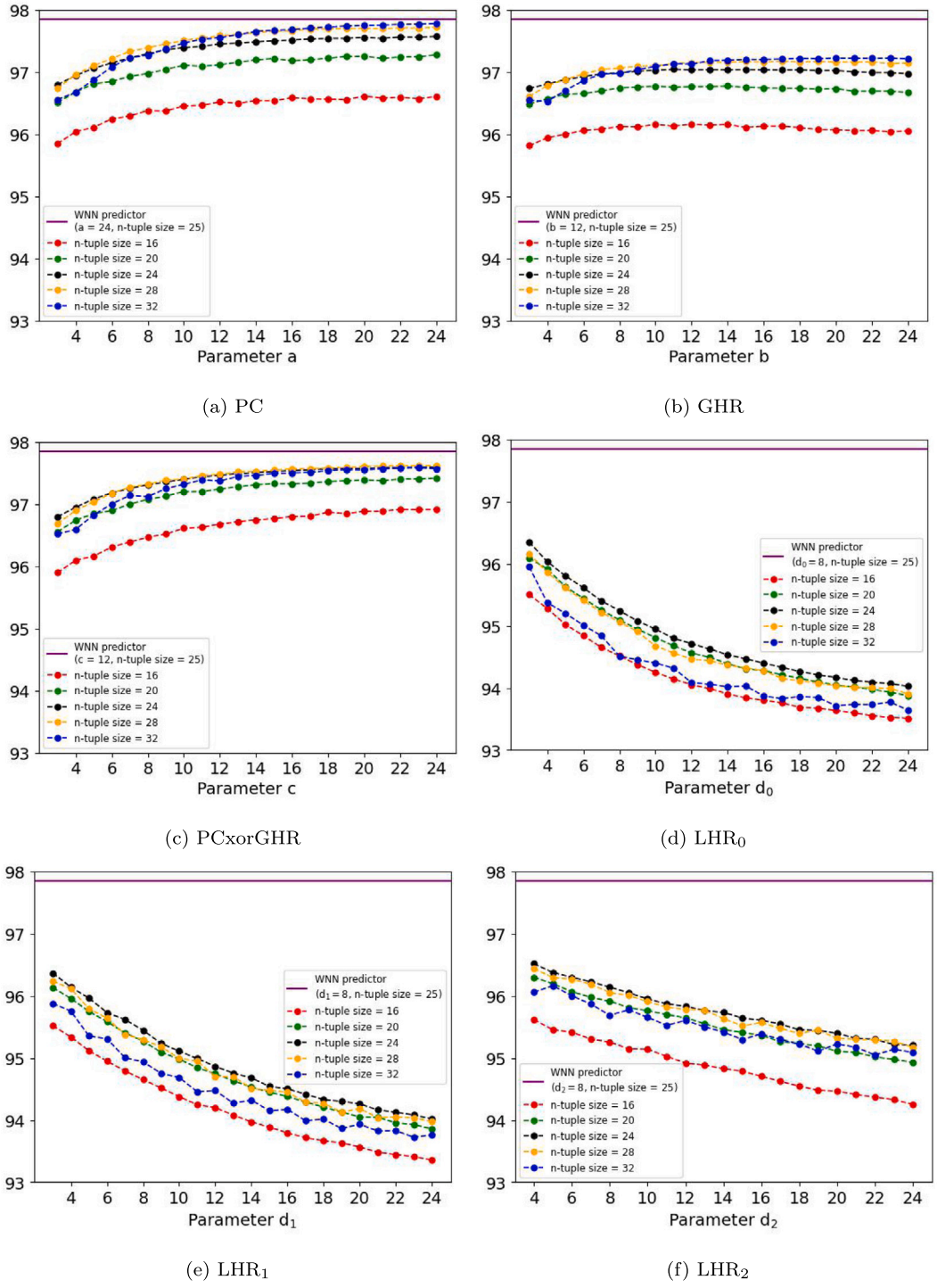
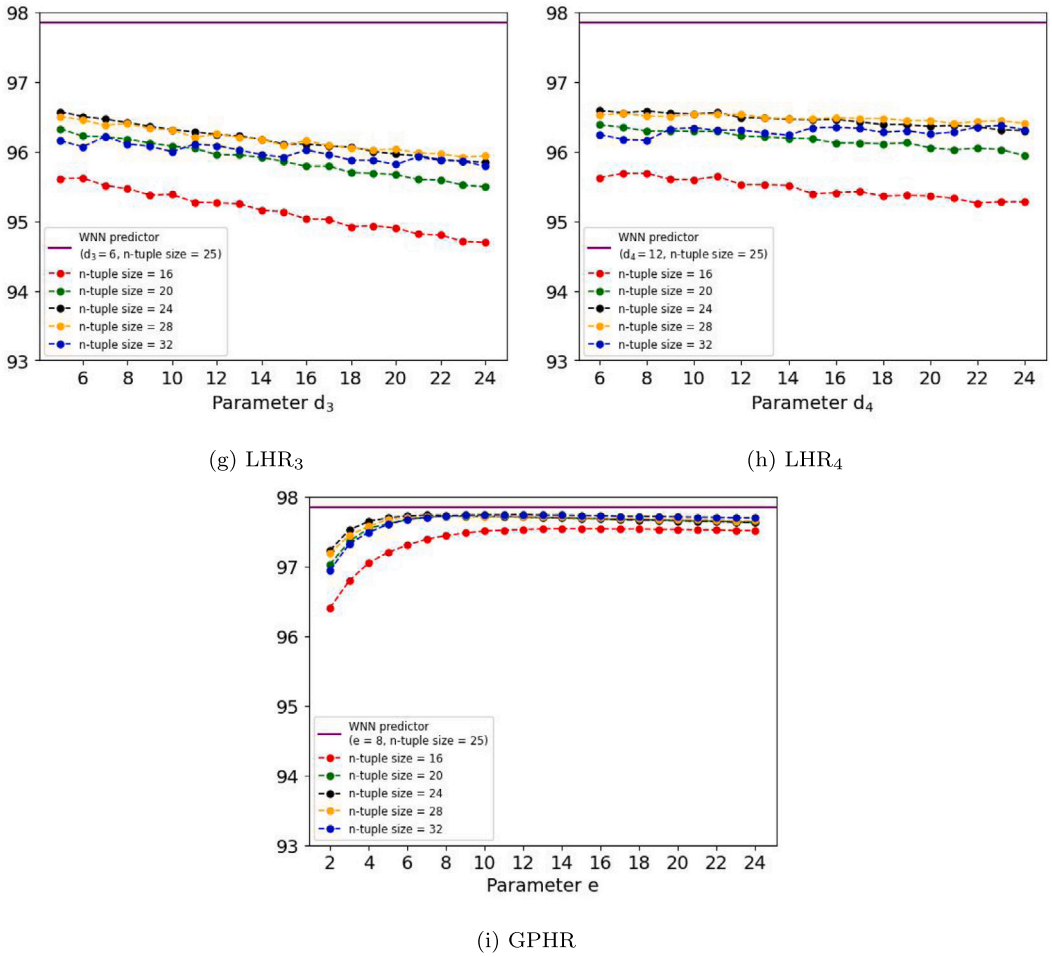


Fig. B.14. Sensitivity Analysis for dataset S2. The most important features are PC and GPHR (Figs. B.14a and B.14i) since the accuracy achieves the highest values when compared to the other features. The accuracy also increases for the features GHR and PCxorGHR (Figs. B.14b and B.14c) but decreases significantly for all other LHR type features (Figs. B.14d–B.14h). In addition, the worst results correspond to n-tuple size 16. Thus, interestingly, the trends of results for both datasets S1 and S2 are somehow similar.

The results show the importance of the parameters, and consequently the corresponding features, for each dataset. In some cases, the accuracy degrades when a particular parameter increases. We hypothesize the reason for this behavior is directly related to the microarchitectural characteristics of the input associated with each benchmark. Furthermore, we observe that as the n-tuple size increases, the accuracy drops significantly depending on the datasets analyzed. This result can be explained by the response of the WiSARD model, which depends on the n-tuple size [27].

In addition, in some cases we observe oscillatory trends of the accuracy curves, especially in the results of datasets I1 and I2 (Figs. B.9 and B.10). This is explained by the fact that binary input size is not a



(g) LHR_3 (h) LHR_4

(i) GPHR

Fig. B.14. (*continued*).

multiple of the n-tuple size in these cases. The WiSARD-based predictor is designed to add "0s" to fill a multiple number of bits of the n-tuple size in the input. Therefore, the standard deviation tends to increase slightly in the experimental results, generating this oscillatory behavior.

References

[1] D.D. Penney, L. Chen, A survey of machine learning applied to computer architecture design, 2019, arXiv preprint arXiv:1909.12373.

[2] D.A. Jiménez, C. Lin, Dynamic branch prediction with perceptrons, in: Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture, IEEE, 2001, pp. 197–206.

[3] D.A. Jiménez, Fast path-based neural branch prediction, in: Proceedings. 36th Annual IEEE/ACM Int Symp on Microarchitecture, 2003. MICRO-36, 2003, pp. 243–252.

[4] A. Smith, Branch prediction with neural networks: Hidden layers and recurrent connections, in: Dept of Comp Science UC, Vol. 92307, San Diego la Jolla, CA, Citeseer, 2004.

[5] S.J. Tarsa, C.-K. Lin, G. Keskin, G. Chinya, H. Wang, Improving branch prediction by modeling global history with convolutional neural networks, 2019, arXiv preprint arXiv:1906.09889.

[6] Y. Mao, H. Zhou, X. Gui, J. Shen, Exploring convolution neural network for branch prediction, IEEE Access 8 (2020) 152008–152016.

[7] P. Michaud, An alternative tage-like conditional branch predictor, ACM Trans. Archit. Code Optim. (TACO) 15 (3) (2018) 1–23.

[8] I. Aleksander, W. Thomas, P. Bowden, WISARD-a radical step forward in image recognition, Sensor Review 4 (3) (1984) 120–124, http://dx.doi.org/10.1108/eb007637.

[9] S.L. Harris, D. Harris, Digital Design and Computer Architecture, RISC-V Ed., Morgan Kaufmann, 2021.

[10] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Elsevier, 2019.

[11] T. Jiang, N. Wu, F. Zhou, L. Zhao, F. Ge, J. Wen, Design of a high performance branch predictor based on global history considering hardware cost, in: 2021 IEEE 4th International Conference on Electronics Technology, ICET, IEEE, 2021, pp. 422–426.

[12] S. Mittal, A survey of techniques for dynamic branch prediction, Concurr. Comput.: Pract. Exper. 31 (1) (2019) e4666.

[13] A. Seznec, P. Michaud, A case for (partially) Tagged geometric history length branch prediction, J. Instruct.-Level Parallel. 8 (2006) 23.

[14] A. Seznec, Tage-sc-l branch predictors, in: JILP-Championship Branch Prediction, 2014, p. 9.

[15] Y. Mao, Z. Huiyang, X. Gui, Exploring Deep Neural Networks for Branch Prediction, ECE Department, NC University, 2017.

[16] D.A. Jiménez, C. Lin, Perceptron learning for predicting the behavior of conditional branches, in: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), Vol. 3, IEEE, 2001, pp. 2122–2127.

[17] D.A. Jiménez, C. Lin, Neural methods for dynamic branch prediction, ACM Trans. Comput. Syst. (TOCS) 20 (4) (2002) 369–397.

[18] G.H. Loh, D.A. Jimenez, Reducing the power and complexity of path-based neural branch prediction, in: Proceedings of the 5th Workshop on Complexity Effective Design, WCED5, 2005, pp. 1–8.

[19] D.A. Jiménez, G.H. Loh, Controlling the power and area of neural branch predictors for practical implementation in high-performance processors, in: 2006 18th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD'06, IEEE, 2006, pp. 55–62.

[20] R.S. Amant, D.A. Jiménez, D. Burger, Low-power, high-performance analog neural branch prediction, in: 2008 41st IEEE/ACM International Symposium on Microarchitecture, IEEE, 2008, pp. 447–458.

[21] D.A. Jiménez, Piecewise linear branch prediction, in: 32nd International Symposium on Computer Architecture, ISCA'05, IEEE, 2005, pp. 382–393.

[22] D.A. Jiménez, Generalizing neural branch prediction, ACM Trans. Archit. Code Optim. (TACO) 5 (4) (2009) 1–27.

[23] D.A. Jiménez, Multiperspective perceptron predictor, in: 5th JILP Workshop on Computer Architecture Competitions: Championship Branch Prediction, CBP-5, 2016, p. 5.

[24] B. Grayson, J. Rupley, G.Z. Zuraski, E. Quinnell, D.A. Jiménez, T. Nakra, P. Kitchin, R. Hensley, E. Brekelbaum, V. Sinha, et al., Evolution of the samsung exynos cpu microarchitecture, in: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture, ISCA, IEEE, 2020, pp. 40–51.
[25] W.W. Bledsoe, I. Browning, Pattern recognition and reading by machine, in: Papers Presented At the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference, 1959, pp. 225–232.
[26] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bull. Math. Biophys. 5 (4) (1943) 115–133.
[27] I. Aleksander, M. De Gregorio, F.M.G. França, P.M.V. Lima, H. Morton, A brief introduction to weightless neural systems, in: ESANN, Citeseer, 2009, pp. 299–305.
[28] N. Spruston, G. Stuart, M. Häusser, Dendritic integration, Dendrites (1999) 231–271.
[29] L.A.L. Filho, L.F. Oliveira, A.L. Filho, G.P. Guarisa, P.M. Lima, F.M. França, Prediction of palm oil production with an enhanced n-tuple regression network, in: ESANN, 2019, p. 6.
[30] B.P. Grieco, P.M. Lima, M. De Gregorio, F.M. França, Producing pattern examples from "mental" images, Neurocomputing 73 (7–9) (2010) 1057–1064.
[31] D. Shkadarevich, Branch prediction, 2020, https://www.kaggle.com/dmitryshkadarevich/branch-prediction.
[32] T.A. Khan, M. Ugur, K. Nathella, D. Sunwoo, H. Litz, D.A. Jiménez, B. Kasikci, Whisper: Profile-guided branch misprediction elimination for data center applications, MICRO, IEEE, 2022, pp. 19–34.
[33] L. Santiago, L. Verona, F. Rangel, F. Firmino, D.S. Menasché, W. Caarls, M. Breternitz Jr., S. Kundu, P.M. Lima, F.M. França, Weightless neural networks as memory segmented bloom filters, Neurocomputing 416 (2020) 292–304.
[34] S. McFarling, Combining branch predictors, Tech. rep., Citeseer, 1993.
[35] T.-Y. Yeh, Y.N. Patt, A comparison of dynamic branch predictors that use two levels of branch history, in: Proceedings of the 20th Annual International Symposium on Computer Architecture, 1993, pp. 257–266.
[36] D. Jiménez, Idealized piecewise linear branch prediction, J. Instr.-Level Parallelism 7 (2005) 1–11.

Luis A.Q. Villon is a master's student in System Engineering and Computer Science at Federal University of Rio de Janeiro (UFRJ). Currently, his research is associated with weightless neural networks and machine learning techniques applied to computer architecture, specially branch prediction. In addition, Luis is a B.S. student in physics at Federal Fluminense University (UFF). In the future, his main interest is to explore the convergence in the fundamentals of quantum mechanics, artificial intelligence and computer architecture.

Zachary Susskind is a Ph.D. student in Computer Engineering at The University of Texas at Austin, where he received his B.Sc. in 2019. His research interests include algorithm-hardware co-design, efficient machine learning, and other topics related to deploying AI in resource constrained "edge" environments.

Alan T.L. Bacellar is a bachelor's student in Applied Mathematics at Federal University of Rio de Janeiro (UFRJ), advised by Felipe M. G. França and Priscila M.V. Lima. His research interests include artificial intelligence, weightless neural networks, deep learning, distributed algorithms and optimization.

Igor Miranda obtained the B.Eng (2007) and M.Eng (2009) degrees in Electronic Engineering and a Ph.D. in Industrial Engineering (2017) from the Federal University of Bahia, Brazil. He joined the Division of Electrical and Computer Engineering, Federal University of Recôncavo da Bahia, as a lecturer in 2013, where he has been assistant professor since 2018. From 2018 to 2019, during a sabbatical year, he was a postdoctoral research associate at the University of Stellenbosch. Since 2022, he has been a visiting researcher at the University of Texas at Austin, supported by the Fulbright program. His research interests lie in the areas of signal processing, machine learning and VLSI design.

Leandro S. de Araújo is Professor of Computer Science and Information System at the Universidade Federal Fluminense (UFF), Brazil. He obtained his D.Sc. (2019) and M.Sc. (2016) in Systems and Computer Engineering at COPPE, Federal University of Rio de Janeiro (UFRJ), Brazil and B.Sc. in Computer Science from the State University of Rio de Janeiro (2014). He has experience in the IT industry with the development of applications on the web and desktop environments. His research interests include Cognitive Architecture, Hardware-Assisted Security, Machine Learning, Reconfigurable Computing and Parallel Programming.

Priscila M.V. Lima B.Sc. in Computer Science from Federal University of Rio de Janeiro (UFRJ) (Magna cum Laude, 1982), M.Sc. in Systems Engineering and Computer Science from COPPE/UFRJ (1987), and her Ph.D. from the Department of Computing, Imperial College London, U.K. (2000). She holds the Chair of Artificial Intelligence of the Brazilian College of Advanced Studies (CBAE) of the UFRJ, Brazil, as a Professor at Tercio Pacitti Institute, and Professor at the Systems Engineering and Computer Science Program, COPPE, UFRJ, Brazil. She has research and teaching interests in artificial intelligence, artificial neural networks, computational intelligence, weightless neural networks, computational logic, distributed algorithms and other aspects of parallel and distributed computing.

Mauricio Breternitz Jr. is a Principal Investigator at Iscte Instituto Universitário de Lisboa. He received the Electronics Engineer degree with honors at ITA-Instituto Tecnologico de Aeronautica, Brazil, a MSc in Computer Science at UNICAMP, Brazil and the Ph.D. in Computer Engineering at Carnegie-Mellon University. He worked at IBM (TJWatson and Austin), Motorola, Intel Labs, TimesN Systems, and AMD Research in Austin,TX. Previously, Mauricio conceived and pushed through deployment innovative algorithmic & microarchitectural ideas that have had significant positive product impact. Mauricio worked on exascale system proposals (part of the AMD Research contribution to the U.S.Department of Energy Exascale Program), on novel algorithms utilizing CPU and GPUs accelerating machine learning, on system-level and architectural-level characterization of cloud workloads and on novel approaches to utilizing CPU and GPU. Mauricio holds 56 U.S. patents and has 55 more pending.

Lizy K. John holds the Truchard Foundation Chair in Engineering in the Department of Electrical & Computer Engineering at The University of Texas at Austin. Her research is in the areas of computer architecture, multi-core processors, memory systems, performance evaluation and benchmarking, workload characterization, and reconfigurable computing. Prof. John's research has been supported by the United States National Science Foundation, Semiconductor Research Consortium (SRC), DARPA, Lockheed Martin, AMD, ARM, Meta, Ampere, Oracle, Huawei, IBM, Intel, Motorola, Freescale, Dell, Samsung, Texas Instruments, etc. She is recipient of NSF CAREER award, UT Austin Engineering Foundation Faculty Award, Halliburton, Brown and Root Engineering Foundation Young Faculty Award, University of Texas Alumni Association Teaching Award, The Pennsylvania State University Outstanding Engineering Alumnus Award, etc. Lizy John holds 15 U. S. patents and has published four books, 16 book chapters, 300+ refereed journal and conference publications, and more than 50 workshop papers. Prof. John is the Editor-in-Chief of IEEE Micro, and has served in the editorial boards of IEEE Transactions on Computers, IEEE Transactions on VLSI, IEEE Transactions on Sustainable Computing, IEEE Computer Architecture Letters, ACM Transactions on Architectures and Code Optimization. She is an IEEE Fellow, ACM Fellow, and Fellow of the National Academy of Inventors.

Felipe M.G. França is a Researcher at the Instituto de Telecomunicações, Universidade do Porto, Portugal, and an Invited Full Professor of Computer Science and Engineering, COPPE, Universidade Federal do Rio de Janeiro (UFRJ), Brazil. He received his Electronics Engineer degree from UFRJ (1982), the M.Sc. in Computer Science from COPPE/UFRJ (1987), and his Ph.D. from the Department of Electrical and Electronics Engineering of the Imperial College London, U.K. (1994). He has published over 250 scientific papers and 2 granted patents. He has experience in Computer Science and Electronics Engineering, acting on the following subjects: artificial neural networks, computational intelligence, weightless neural networks, computer architecture, cryptographic circuits, dataflow computing, distributed algorithms, collective robotics, intelligent transportation systems.

Diego L.C. Dutra is currently a professor at the Federal University of Rio de Janeiro (UFRJ), Brazil, where he is a member of the COMPASS Laboratory and leader of the HEADS research group. He worked as a postdoctoral researcher in the MOSAIC Lab until 2017. He received his D.Sc. in Systems Engineering and Computer Science Program from UFRJ in 2015. His research interests include computer architecture, high-performance computing, virtualization, cloud computing, microarchitecture security, mobile systems, and Software-Defined Systems.