A CONDITIONAL BRANCH PREDICTOR BASED ON WEIGHTLESS NEURAL NETWORKS

Luis Armando Quintanilla Villon

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Diego Leonel Cadette Dutra
Felipe Maia Galvão França

Rio de Janeiro
Setembro de 2023

A CONDITIONAL BRANCH PREDICTOR BASED ON WEIGHTLESS NEURAL NETWORKS

Luis Armando Quintanilla Villon

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Diego Leonel Cadette Dutra
Felipe Maia Galvão França

Aprovada por: Prof. Diego Leonel Cadette Dutra
Prof. Claudio Luis de Amorim
Profa. Ana Cristina Costa Aguiar

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2023

Villon, Luis Armando Quintanilla

A conditional branch predictor based on weightless neural networks/Luis Armando Quintanilla Villon. – Rio de Janeiro: UFRJ/COPPE, 2023.

XV, 49 p.: il.; 29, 7cm.

Orientadores: Diego Leonel Cadette Dutra
Felipe Maia Galvão França

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2023.

Referências Bibliográficas: p. 31 – 34.

1. Branch predictor. 2. Weightless Neural Networks. 3. WiSARD. I. Dutra, Diego Leonel Cadette *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao primórdio indefinido de todas
as coisas.

Agradecimentos

Gostaria de agradecer em primeiro lugar à minha esposa, Lilian, por sua dedicação, suporte e ajuda incontrastáveis e incomparáveis, sem os quais este trabalho não teria sido possível.

Aos meus pais, Rosa e Washington, por incutir em mim a importância da continuidade do ensino superior e por sempre buscar a excelência.

Aos meus orientadores, Diego e Felipe, pela compreensão, paciência e por me guiar nos meus primeiros passos no mundo da pesquisa acadêmica de alto nível. Igualmente gostaria de agradecer à professora Priscila que junto ao professor Felipe me guiaram neste processo com paciencia e dedicação.

Ao PESC, por me permitir participar desse desafio abrindo as portas e me dando oportunidade de crescimento profissional em benefício da sociedade.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM PREDITOR DE DESVIO CONDICIONAL BASEADO EM REDES NEURAIS SEM PESO

Luis Armando Quintanilla Villon

Setembro/2023

Orientadores: Diego Leonel Cadette Dutra
Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

A previsão de desvio condicional permite a busca especulativa e a execução de instruções antes de saber a direção de instruções condicionais. Como em outras áreas, as técnicas de aprendizado de máquina são uma abordagem promissora para a construção de preditores de desvio, como por exemplo, o preditor Perceptron. No entanto, essas soluções tradicionais exigem grandes tamanhos de entrada, o que impõe uma considerável sobrecarga de área. Esta dissertação propõe um preditor de desvio condicional baseado no modelo de rede neural sem peso WiSARD (Wilkie, Stoneham e Aleksander's Recognition Device). O preditor baseado em WiSARD implementa one-shot online training projetado para abordar a previsão de desvio como um problema de classificação binária. Este trabalho compara o preditor baseado em WiSARD com dois preditores do estado da arte: TAGE-SC-L (TAgged GEometric - Statistical Corrector - Loop) e o Multiperspective Perceptron. A avaliação experimental mostra que o preditor proposto, com um tamanho de entrada menor, supera o preditor baseado em perceptron em cerca de 0,09% e atinge precisão semelhante à do TAGE-SC-L. Além disso, foi realizada uma análise de sensibilidade experimental para encontrar o melhor preditor para cada conjunto de dados e, com base nesses resultados, projetaram-se novos preditores especializados usando uma composição de parâmetros específica para cada conjunto de dados. Os resultados mostram que o preditor especializado baseado em WiSARD supera o estado da arte em mais de 2,3% no melhor caso. Ademais, por meio da implementação de classificadores de preditores especializados, descobriu-se que utilizar 90% do preditor especializado para um conjunto de dados específico rendeu desempenho comparável ao preditor especializado correspondente.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A CONDITIONAL BRANCH PREDICTOR BASED ON WEIGHTLESS NEURAL NETWORKS

Luis Armando Quintanilla Villon

September/2023

Advisors: Diego Leonel Cadette Dutra
Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

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. This dissertation proposes 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. This work compares the WiSARD-based predictor with two state-of-the-art predictors: TAGE-SC-L (TAgged GEometric - Statistical Corrector - Loop) and the Multiperspective Perceptron. The experimental evaluation shows that the 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, an experimental sensitivity analysis was performed 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.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN Artificial Neural Networks, p. 10

BHR Branch History Register, p. 7

BHT Branch History Table, p. 6

CBP-3 3^{rd} Championship Branch Prediction, p. 17

FSM Finite State Machine, p. 6

GHR Global History Register, p. 7

GPHR Global Path History Register, p. 13

ILP Instruction Level Parallelism, p. 4

LHR Local History Register, p. 13

LHT Local History Table, p. 14

LUTs Lookup Tables, p. 1

PCxorGHR xor operation for the PC and GHR, p. 13

PC Program Counter, p. 5

PHT Pattern History Table, p. 7

RAM Random Access Memory, p. 1

TAGE-SC-L TAgged GEometric - Statistical Corrector - Loop, p. 2

TAGE TAgged-GEometric, p. 10

WNNs Weightless neural networks, p. 1

WiSARD Wilkie, Stoneham, and Aleksander's Recognition Device, p. 1

Chapter 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 Random Access Memory (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] was the first weightless neural model to achieve commercial success and is the neural network model adopted in this dissertation.

1.1 Motivation

Due to the ability to learn non-linear features indirectly represented by the inputs and the relatively simple arithmetic operations, the WiSARD model is an attractive alternative to traditional neural-based predictors. Nevertheless, as far as we 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 of using a WiSARD-based branch predictor, for which we propose a new predictor architecture.

1.2 Goals and contribution

As the main goal of this dissertation, the novel predictor approach based on weightless neural networks was designed to treat branch prediction as a binary classification problem. This work also explores how the proposed predictor performs one-shot online training methodology.

In this dissertation, we perform four experimental approaches. First, by making a pseudo-exhaustive hyperparameters search, we found a WiSARD-based predictor configuration that requires smaller input sizes when compared to state-of-the-art predictors, the TAgged GEometric - Statistical Corrector - Loop (TAGE-SC-L) and the Multiperspective perceptron. These results show that the WiSARD-based predictor can achieve similar accuracy and even outperform the other two predictors, depending on the analyzed dataset. On average, the WiSARD-based predictor is tied in accuracy with TAGE-SC-L, and outperforms the Multiperspective perceptron by approximately 0.09%.

Next, we performed a sensitivity analysis in order to explore the most important input fields, which directly leads to find the best predictor parameters configuration for each dataset. Then, we compared the best results of these specialized predictors with another three predictors, namely: The first result with the first input composition, the TAGE-SC-L and the Multiperspective perceptron. This comparison shows that a particular specialized WiSARD-based predictor outperforms the state of the art by more than 2.3% in the best case. Furthermore, for all cases, the best specialized predictor configuration for each dataset exceeds the first preliminary corresponding result.

Lastly, by employing dedicated predictor classifiers, this work revealed that using 90% of the specialized predictor for a particular dataset resulted in performance comparable to that of the corresponding specialized predictor.

1.3 Structure of the text

The rest of this dissertation is organized as follows. Chapter 2 presents the background and fundamental concepts related to branch prediction, the most important branch predictors historically used in the industry and weightless neural networks. Chapter 3 describes the proposed WiSARD-based predictor architecture as well as how its input is composed. Chapter 4 presents the experimental data and the

methodology used. Chapter 5 shows the experimental results and the main discoveries are discussed in detail. Finally, Chapter 6 concludes this work. In addition, Appendix A shows the full results of the sensitivity analysis described in Chapter 5.

Chapter 2

Background

This Chapter provides the relevant background on fundamental concepts related to conditional branch prediction, weightless neural networks and the WiSARD model. In particular, section 2.4 exhibits the main type of WNNs employed to construct the proposed predictor architecture in this dissertation.

2.1 Conditional branches

Conditional branches are usually employed to make decisions in the execution of a program, based on comparison between logical expressions. In high level programming languages, conditional branches are usually expressed by *if* or *if-else* statements. From the computer architecture perspective, the compiler interprets the conditional branches statements to conditional branch instructions. This instructions modify the flow of the program so that the processor can fetch instructions that are not in sequential order in memory [9], which lead to performance issues.

2.1.1 Instruction-Level parallelism and branch hazards

In order to avoid bubbles and stalls due to conditional branch instructions, nearly all modern processors are pipelined in several stages to overlap the execution of instructions and improve performance. Because instructions can be evaluated in parallel, this mechanism among instructions is called Instruction Level Parallelism (ILP) [10] .

In general, to establish how much parallelism exists and if this parallelism can be exploited in the flow of a program, it is necessary to know how the instructions depend on each other. Specifically, for branch instructions it is critical to determine the ordering of instructions after the branch because every instruction in the branch block must be executed after the the instruction preceded by the branch. The control of the flow of the instructions in a pipelined processor is called *control dependence*. A

Branch Hazard occurs when the proper instruction after a branch in the instruction stream can not execute during its designated clock cycle due because the instruction that was fetched is not the one that is needed [11].

Branch hazards can cause a greater performance loss in a pipelined processor, reinforced by the fact that branch instructions are frequent, composing nearly 20-30% of all instructions of computer programs [12]. When a branch is executed, it may or may not change the Program Counter (PC) to something other than its current value plus 4. If a branch changes the PC to its target address, it is a taken branch; otherwise, it is not taken [10].

2.1.2 Branch prediction

A method of resolving a branch hazard is to predict a given outcome for the conditional branch and proceeds from that assumption rather than waiting to ascertain the actual outcome [11]. In most applications, branches are usually highly predictable, and fortunately, the vast majority of programs have branches with this characteristic most of the time through the analysis of some of its relevant features [13]. The technology dedicated to solve branch hazard using some type of prediction strategy is called *branch prediction*

The simplest method of predicting branches is by analyzing only the data collected from earlier runs provided by profile-based information [10]. This type of prediction is called *static branch prediction* and in most cases the instructions do not change. This technique are useful for some specialized applications in which branches can be predicted with a medium-high or tolerable accuracy at compile time, such as in scientific and floating-point programs [13].

Static branch prediction is based on stereotypical behavior and does not take into account some characteristics of conditional branches, such as the individuality of a specific branch instruction which may depend on particular conditions. In contrast, *dynamic branch prediction* makes a prediction depending on the demeanor of each conditional branch and can change predictions for a conditional branch over the entire execution of a program [11]. In order to increase the accuracy by implementing a dynamic branch prediction technique, a processor explores and takes advantage of current execution patterns, which can come from different micro-architectural data sources.

Dynamic branch predictors for conditional branches are the focus of this dissertation, and several of the main related techniques are mentioned in Section 2.2.

2.2 Conditional branch predictor

Nearly all modern processors implement a conditional branch predictor unit in its microarchitecture design [10]. Instead of stopping when a conditional branch is encountered in the execution of a program, a processor employs a conditional branch predictor to fetch and speculatively execute instructions along a predicted path. The main idea consist in predict branches based on dynamic information provided by the behavior coming from the lowest layer in the microarchitecture level.

In general, a branch predictor is composed of three elements: 1) an input, which carries information and features about the current instruction; 2) the prediction, which is going to notify the system if the current branch will be taken or not; 3) and the main predictor architecture, which uses arithmetic or logic operations to perform the prediction.

Furthermore, as computer architectures become more complex and the number of instructions issued per cycle increases, the penalty for a prediction error (misprediction) increases [2]. Since branch misprediction can result in both high latency and high energy consumption, a minimal improvement in branch prediction accuracy can boost performance and energy efficiency significantly, as indicated in related work [14] [5] [15].

The following subsections describe in detail the most important conditional branch prediction techniques.

2.2.1 Branch history table

By far, the simplest way to implement a dynamic branch predictor is to index the lower portion of some information that represents the conditional branch [10]. This information can be taken from the address of the branch instruction directly. The most common implementation is a 2-bit prediction scheme [16], which can be modeled as a Finite State Machine (FSM) . Finite state machines or finite automata are the most simple computational model and are useful tools for recognizing patterns in data [17]. In particular, the 2-bit saturation counter predictor is a Moore State Machine because its output depends on the state of the system [9]. The design of a 2-bit FSM is shown in Figure 2.1. There are four states indexed using two-bit counters: strong taken (11), weak taken (10), weak not taken (01), and strong not taken (00). Weak states are related to the current state transition, that is, how easy it is to change the state. The state is updated depending on the real direction of its corresponding conditional branch.

A Branch History Table (BHT) is a memory indexed by the lower portion of the address of the branch instruction (Figure2.2). When using a 2-bit prediction scheme, the BHT contains 2 bits in each row in the table that notify whether the

branch was previously taken or not. The outcome of the prediction will be based on the position of the state machine. It will predict taken if the state is (11) or (10), otherwise and it will predict not taken.

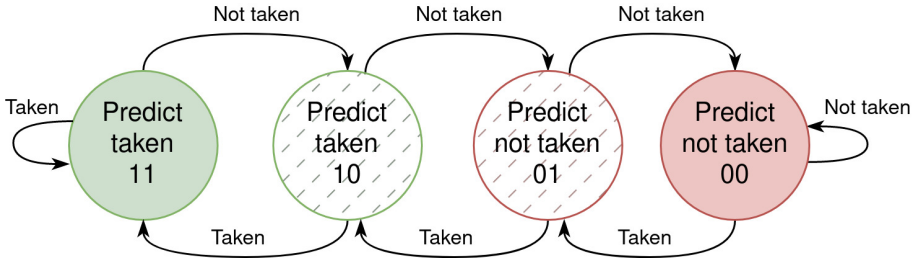


Figure 2.1: A state diagram of 2-bit saturation counter modeled as a FSM. Each state is associated with 2 bits. These 2 bits encode the four states in the system. The weakly taken and not taken states are represented with dashed lines.

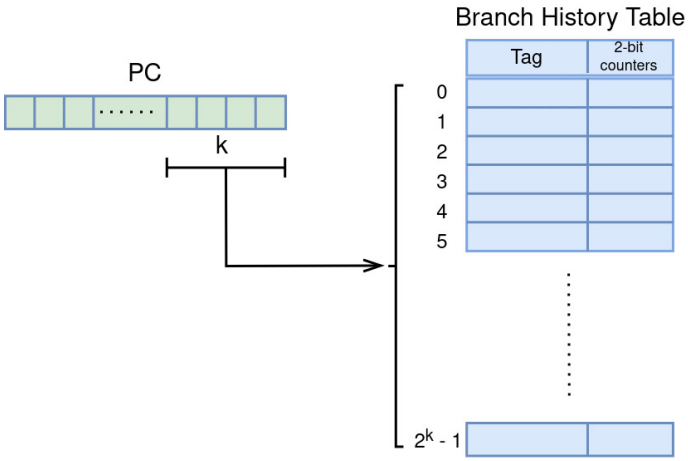


Figure 2.2: A representation of the BHT. Each entry of the BHT consists of the address of a branch instruction and a two-bit saturating counter.

2.2.2 Two-level adaptive predictor

An important disadvantage of the BHT is that it only uses the recent behavior of a particular conditional branch in order to predict the future outcome of that branch. A possible way to improve the accuracy is to look at the recent behavior of other conditional branches from the execution of the program.

In general, conditional branch predictors that use the information of other branches outcomes are called two-level adaptive predictors or correlating predictors [10] [18]. This type of predictors stores different branch histories in two levels of memory called Branch History Register (BHR) and branch Pattern History Table (PHT) [19] [20] . The BHR is a table somehow similar to the BHT from the 2-bit prediction scheme. The BHR is also called Global History Register (GHR)

This branch predictor is highly reconfigurable and there are nine possible configurations 2.1. In GAx schemes, the first level is represented by a single GHR, which is a shift register that keeps the actual last k branches encountered (Figure 2.3). Meanwhile, in SAx schemes, the first level consists in a BHR table that stores the last occurrences of the conditional branch instructions from the same subset (Figure 2.4). On the other hand, in PAx schemes the first level conrresponds to information of the same branch instruction (Figure 2.5), thus, one history register is associated with each conditional branch [20]. In all two-level predictor variations, the second level keeps the information in a PHT table.

Name (Variation)	First Level (Branch History)	Second Level (Pattern History)
GAg	Kept Globally	Kept Globally
GAs	Kept Globally	Kept per Set
GAp	Kept Globally	Kept per Address
SAg	Kept per Set	Kept Globally
SAs	Kept per Set	Kept per Address
SAp	Kept per Set	Kept per Set
PAg	Kept per Address	Kept Globally
PAs	Kept per Address	Kept per Address
PAp	Kept per Address	Kept per Set

Table 2.1: Variations of the Two-level Adaptive Predictor

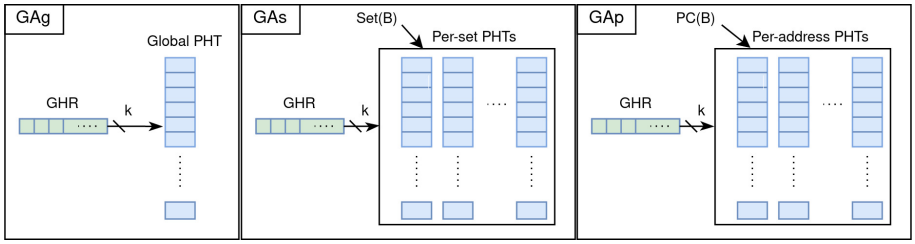


Figure 2.3: The three types of global history predictors (GAx).

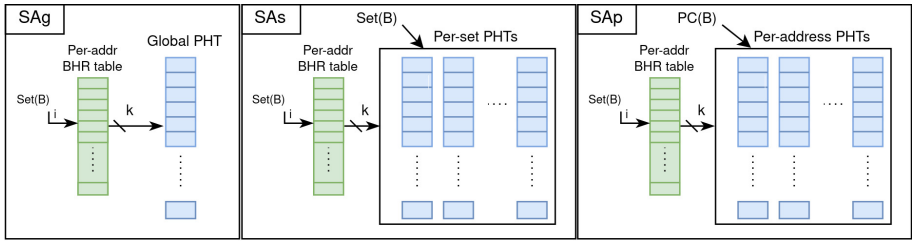


Figure 2.4: The three types of per-set history predictors (SAx).

2.2.3 The gshare predictor

In order to reduce aliasing in a two-level predictor which uses global history of conditional branches, the gshare predictor brings the best of global history and

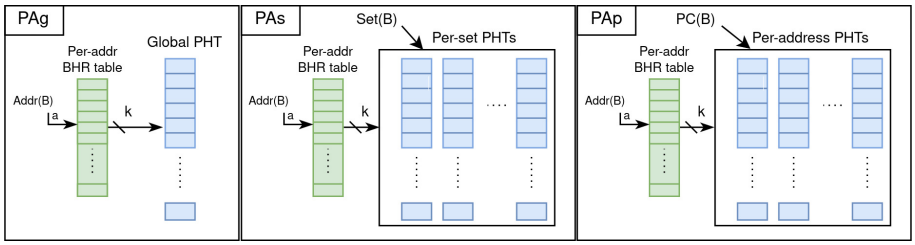


Figure 2.5: The three types of per-address history predictors (PAx).

branch information together [21] [14]. The strategy consist in apply the exclusive disjunction, or *xor* operation, between the PC and the GHR. Then, the result is indexed in order to get or update information in a table of 2-bit counters (Figure 2.6), similar to the BHT from the 2-bit predictor scheme.

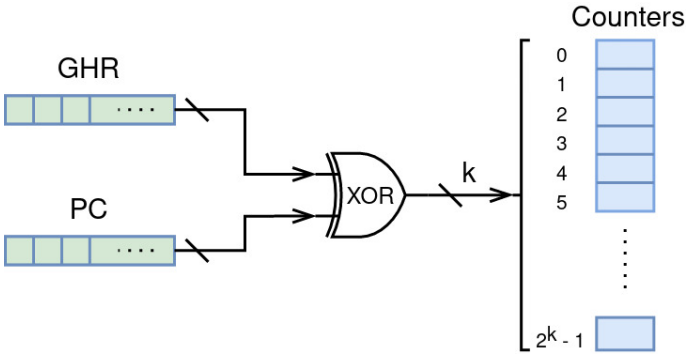


Figure 2.6: Architecture of the gshare predictor

2.2.4 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 [22] [23] [3]. Research projects to reduce the power consumption, complexity [24] [25] [26], and also, to deal with the impossibility, inherent to perceptron models, to learn nonlinear functions from the inputs [27] [28] 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 perspectives [29]. This predictor uses pattern histories and features based on other metrics, which results in large input sizes for the data to be linearly separable (Figure 2.7). 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 can be useful in branch prediction for industrial applications [30].

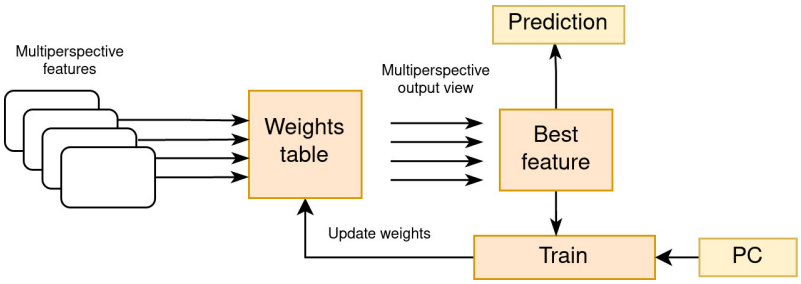


Figure 2.7: Main architectural view of the Multiperspective Perceptron predictor. This hashed predictor that uses not only global path and pattern histories, but also other features and metrics.

2.2.5 The TAGE-SC-L predictor

Most modern branch predictors are variants of the TAgged-GEometric (TAGE) [31] and/or perceptron branch predictors [2]. In particular, the TAGE-SC-L [32] predictor is considered the state-of-the-art in the industry [33] and it won the last branch predictor championship celebrated in 2016. In this predictor, the input consist of a large global history register and other microarchitecture features (Figure 2.8). 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 a neural-based statistical corrector is implemented to detect some unlikely predictions and to revert them.

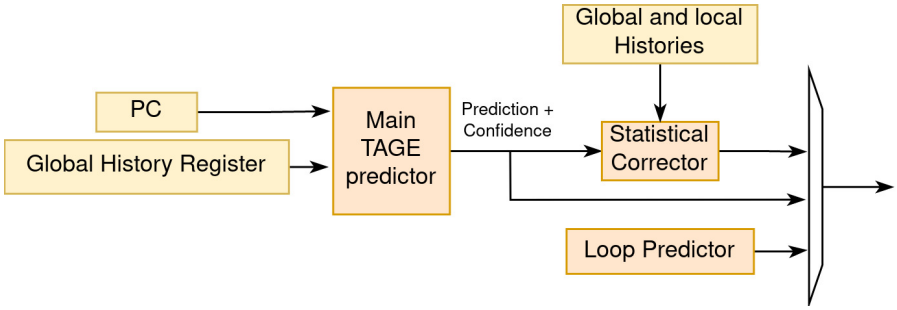


Figure 2.8: Depiction of the TAGE-SC-L predictor. It consists on a TAGE predictor backed with a loop predictor and a statistical corrector

2.3 Weightless neural networks

WNNs were developed by BLEDSOE e BROWNING [34] who named it as *n-tuple classifier*. Similarly to traditional Artificial Neural Networks (ANN) in their early days [35], WNNs were initially inspired by the human nervous system. However, in WNNs the dendritic tree is prioritized, unlike conventional ANN paradigms which are based on weighted-sum-and-threshold neurons [36]. This is an important fact since the vast majority of synapses terminate on the neuron's dendritic tree [37].

In the *n-tuple classifier*, the nodes are based on RAM where the information learned is stored. The functionality of a neuron can be modified by changes in the RAM contents. In contrast to weighted-sum-and-threshold artificial neurons, WNNs can directly map exclusive-OR functions on a n-tuple RAM node [38].

2.3.1 RAM-Discriminators

A Discriminator is a set of N RAM nodes which have n address lines each. Thus, the input of a RAM-discriminator is a binary pattern of $N \cdot n$ bits. A biunivocal pseudo-random mapping is established between the RAM addresses lines and the the input pattern (Figure 2.9). The *n-tuple classifier* randomly chooses groups of n binary positions of an input and use them to address what is called Address Groups, each tuple addressing one group. Moreover, *n-tuple classifier* implements a summing device in order to obtain an input in a classification process.

2.4 WiSARD

The WiSARD was the first weightless neural network distributed commercially. 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 [39] and is trained on a particular class of patterns.

To illustrate how the WiSARD model works, Figure 2.10 describes an example implemented for digit recognition tasks, applied to a binarized image in a matrix representation. 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. The index of the discriminator with the highest response value is taken as the predicted class. To deal with the problem of

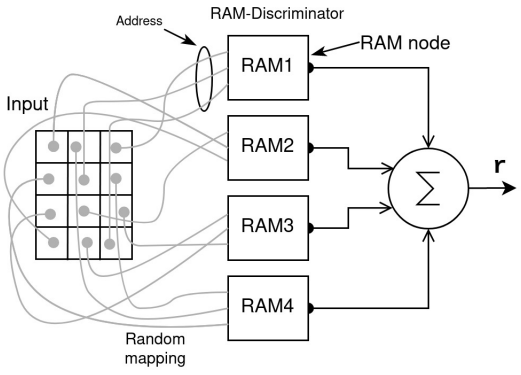


Figure 2.9: The RAM-discriminator basic structure

learning saturation, the contents of the RAM nodes are implemented as an access counter which 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 [40] 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, Figure 2.11 presents the structure of RAMs in a Discriminator.

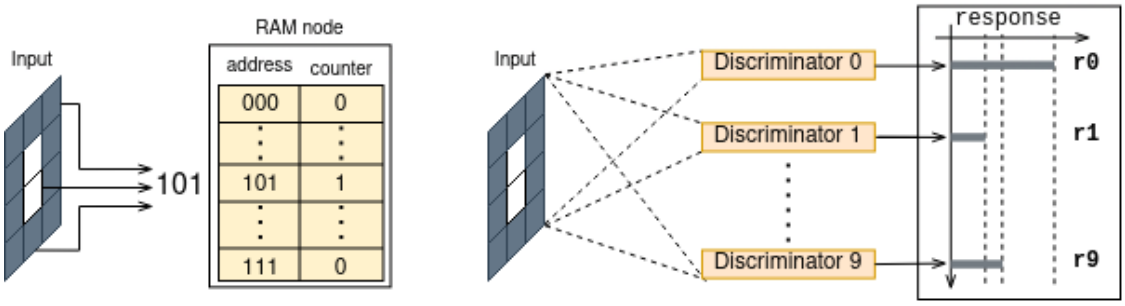


Figure 2.10: In this representation of the WiSARD model example, the input image contains "0". It shows an outline of the training phase on the left side. On the right side, the corresponding discriminator produces the strongest response in the classification phase.

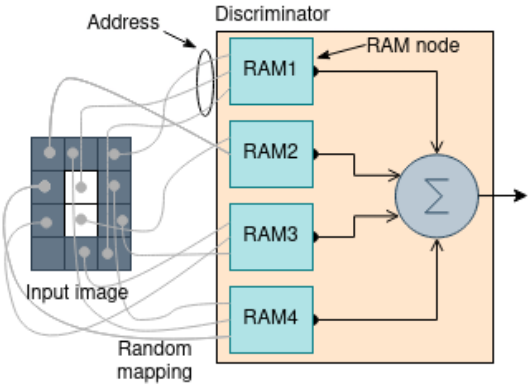


Figure 2.11: A representation of the RAM-Discriminator structure.

Chapter 3

Proposal: WiSARD-based conditional branch predictor

This dissertation proposes a novel predictor architecture based on the WiSARD model, which can be modified depending on the constraints of the applications. The WiSARD-based predictor is designed to perform one-shot online training and the respective classification phase performs a binary classification. This chapter illustrates, in Sections 3.1 and 3.2 respectively, how the input is composed and the entire predictor architecture.

3.1 Input composition

The binary input is a linear combination of different sources of current and recent branch address information. We choose the features inspired by previous work related to conditional branch techniques shown in Section 2.2. These features are the following: Some bits of the Program Counter (PC) which represents the least significant bits from the current branch address instruction, Global History Register (GHR) from the last conditional branches outcomes, the xor operation for the PC and GHR (PCxorGHR) , Local History Register (LHR) from the current branch and the Global Path History Register (GPHR) which stores the 8 less significant bits from the last 8 conditional branches. Thus, the input can be expressed by the relation:

$$input = a \cdot PC + b \cdot GHR + c \cdot PCxorGHR + \sum_{i=0}^{N-1} d_i \cdot LHR_i + e \cdot GPHR \quad (3.1)$$

The additional parameters *a, b, c,* d_i and e represent the strength of its associated feature in the input. This input composition assumes that there are N LHR of different sizes. The following paragraphs describe these features through a more

complete and detailed explanation.

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

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.

XOR operation between PC and GHR (PCxorGHR). As mentioned and previously addressed in related works [21], 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.

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 [20]. Figure 3.1 illustrates how the least significant bits of the PC are employed to update the LHR in the LHT.

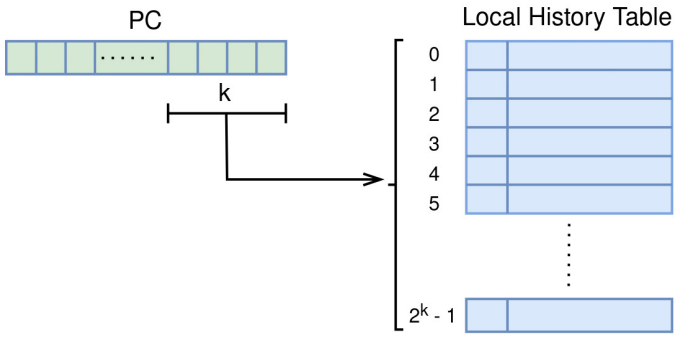


Figure 3.1: Structure and composition of a LHT. Each row in the table represent a particular LHR

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 [41] [33].

3.2 Input composition example

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, Figure 3.2 shows how we use the parameters and registers to compose the input. In this example, the input size is 208 bits.

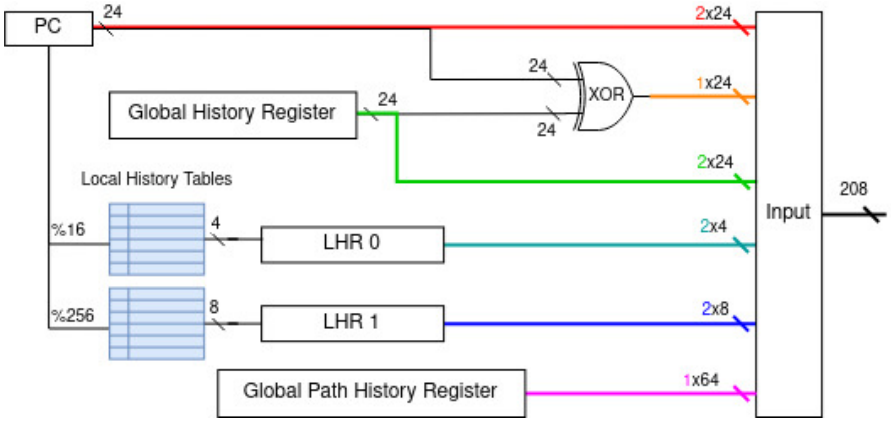


Figure 3.2: A depiction of the input composition. In this example the input size is 208 bits

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

$$input = \{\{2\{PC[23:0]\}\}, \{1\{PC[23:0] \wedge GHR\}\}, \{2\{GHR\}\}, \{2\{LHR0\}\}, \{2\{LHR1\}\}, \{1\{GPHR\}\}\} \quad (3.2)$$

3.3 Predictor architecture

Figure 3.3 shows the WiSARD-based predictor architecture. 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 response from both discriminators, and the one with the highest response value determines the corresponding final output. In addition, the architecture implements 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.

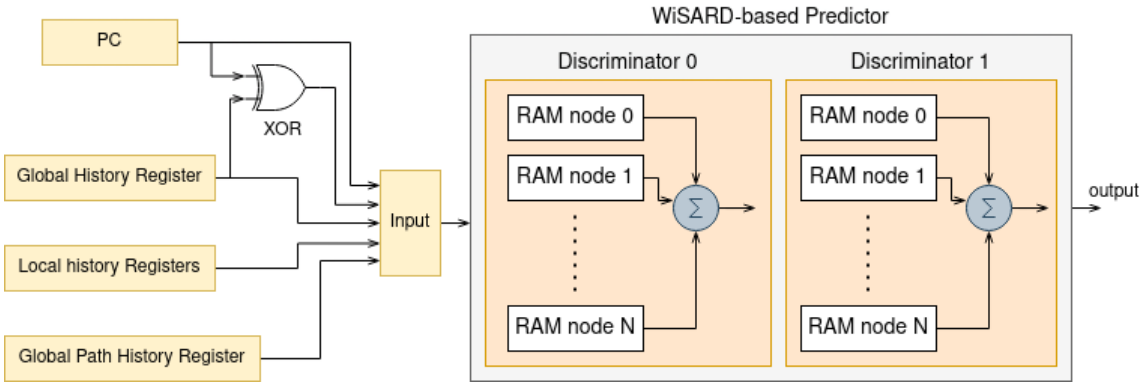


Figure 3.3: A depiction of the WiSARD-based predictor. In this example there are three local history registers

Chapter 4

Methodology and experimental setup

This Chapter describes the datasets and the required experimental setup to evaluate the proposed predictor described in Chapter 3.

4.1 Dataset

This Dissertation uses datasets from the 3^{rd} Championship Branch Prediction (CBP-3) organized by the JILP Workshop on Computer Architecture Competitions, which can be found in Kaggle [42]. 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 $4x10^5$ conditional branch instructions, with the exception of dataset M1, which has $3x10^5$ elements.

Among these datasets, this dissertation only considers the PC and the actual outcome of each branch. These are used to build the microarchitecture information that compounds the input, as described in Chapter 3 and Appendix A. Moreover, as we design the proposed 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

This dissertation 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 mapping and uniform distribution of inputs over RAMs impacts the overall performance of the proposed design. Even though a given

final hardware implementation must have a fixed mapping, this approach allows one to understand how the results depend on the input mapping. We set a fixed size, in bits, for the features in the input from Equation 3.1, 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:

$$input = 24a + 24b + 24c + 24d_0 + 16d_1 + 9d_2 + 7d_3 + 5d_4 + 64e \quad (4.1)$$

In the rest of this dissertation, we will use the Equation 4.1 for all different experimental scenarios. Furthermore, this dissertation compare the proposed 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.

Chapter 5

Results and discussion

This chapter shows 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 the proposed predictor whose accuracy would outperform, on average, the state-of-the-art predictors. It is important to 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, the first 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 the Equation 4.1, 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 Figure 5.1. It illustrates how the accuracy varies as the size of the n-tuple 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.

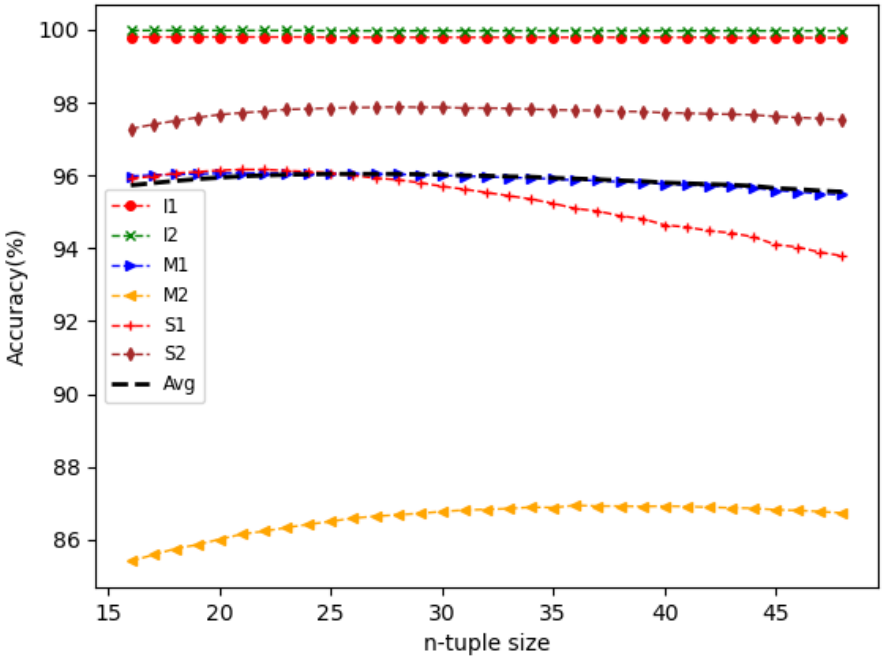


Figure 5.1: Results of accuracy obtained by the WiSARD-based predictor in the classification phase as the size of the n-tuple increases. The "Avg" line represents the average accuracy of all datasets. Higher is better.

This dissertation compares these results with the other predictors mentioned in Chapter 2, including the state-of-the-art, namely the TAGE-SC-L and Multiperspective Perceptron predictors (shown in Tables 5.1 - 5.3 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. Also, the memory size of all predictors except WNN is 64KiB. Undoubtedly, the WNN, T and MP predictors substantially exceeds the others ones in all six datasets (Tables 5.1 - 5.3). In consequence, we only present accuracy results of WNN, T and M predictors in Table 5.4 for better visualization. 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%. It is important to emphasize that the proposed predictor shows a higher accuracy value on the dataset M2 compared to the 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 5.5. Interestingly, the accuracies obtained with these

Predictor	I1	I2
WNN	99.7948±.0016	99.9749±.0010
BHT	98.5892±.0000	98.2440±.0000
GAg	99.0910±.0000	99.1568±.0000
GAs	98.6785±.0000	99.9635±.0000
GAp	98.7087±.0000	99.0348±.0000
SAg	98.5027±.0000	97.3440±.0000
SAs	99.1813±.0000	98.3370±.0000
SAp	99.2002±.0000	97.6135±.0000
PAg	99.2185±.0000	99.9608±.0000
PAs	99.2067±.0000	99.9350±.0000
PAp	99.1990±.0000	99.9585±.0000
Gshare	99.2245±.0000	99.1470±.0000
T	99.8138±.0000	99.9782±.0000
MP	99.7700±.0000	99.9792±.0000

Table 5.1: Accuracy results for datasets I1 and I2 compare the best configurations of the WiSARD-based predictor with other predictors previously mentioned in Chapter 2: Branch History Table, the nine variations of the Two-level Adaptive (Table 2.1), gshare, the TAGE-SC-L and the Multiperspective Perceptron predictors. We labeled them WNN, BHT, GAx, SAx, PAx, Gshare, T and MP, respectively.

Predictor	M1	M2
WNN	96.0540±.0224	86.4968±.1458
BHT	87.1183±.0000	83.6300±.0000
GAg	92.5803±.0000	81.3975±.0000
GAs	93.5330±.0000	82.2255±.0000
GAp	93.2197±.0000	82.7135±.0000
SAg	86.1627±.0000	79.1850±.0000
SAs	88.7113±.0000	79.7140±.0000
SAp	90.0560±.0000	80.7100±.0000
PAg	89.4027±.0000	82.4775±.0000
PAs	91.2670±.0000	83.1155±.0000
PAp	89.8390±.0000	83.2752±.0000
Gshare	92.3827±.0000	79.8717±.0000
T	96.1357±.0000	85.8582±.0000
MP	96.2340±.0000	85.7533±.0000

Table 5.2: Accuracy results for datasets M1 and M2 compare the best configurations of the WiSARD-based predictor with other predictors previously mentioned in Chapter 2: Branch History Table, the nine variations of the Two-level Adaptive (Table 2.1), gshare, the TAGE-SC-L and the Multiperspective Perceptron predictors. We labeled them WNN, BHT, GAx, SAx, PAx, Gshare, T and MP, respectively.

inputs, on average, were 77.1948% and 89.7698% respectively. As expected, the proposed predictor has a completely different knowledge acquisition process than the other predictors since the discrepancy in accuracy is considerable.

Predictor	S1	S2
WNN	96.0651±.0349	97.8504±.0124
BHT	92.6352±.0000	92.4316±.0000
GAg	91.8653±.0000	92.5344±.0000
GAs	92.3012±.0000	93.1333±.0000
GAp	92.6478±.0000	93.1271±.0000
SAg	90.2090±.0000	89.6268±.0000
SAs	91.4223±.0000	90.8722±.0000
SAp	91.8902±.0000	91.3441±.0000
PAg	93.5727±.0000	93.1425±.0000
PAs	93.8405±.0000	93.4029±.0000
PAp	93.1838±.0000	92.5567±.0000
Gshare	94.9577±.0000	92.6609±.0000
T	97.8710±.0000	95.9964±.0000
MP	97.7645±.0000	95.9525±.0000

Table 5.3: Accuracy results for datasets S1 and S2 compare the best configurations of the WiSARD-based predictor with other predictors previously mentioned in Chapter 2: Branch History Table, the nine variations of the Two-level Adaptive (Table 2.1), gshare, the TAGE-SC-L and the Multiperspective Perceptron predictors. We labeled them WNN, BHT, GAx, SAx, PAx, Gshare, T and MP, respectively.

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

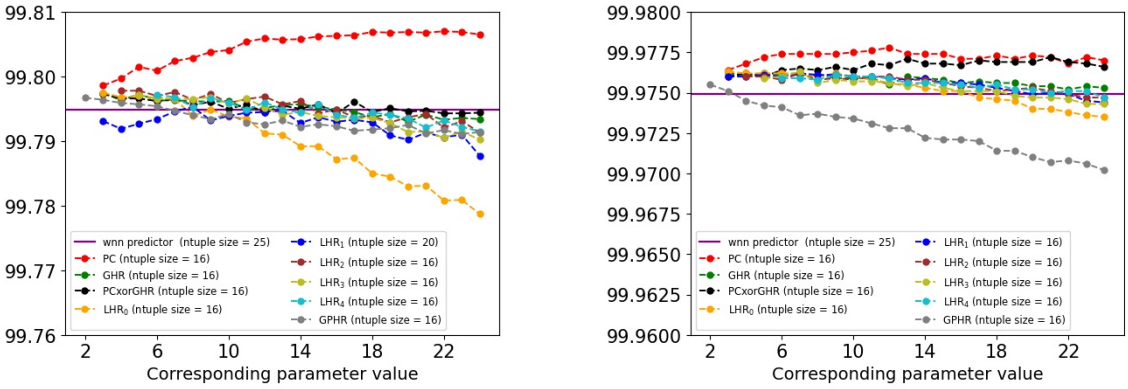
Table 5.4: Accuracy results compare the best configurations of the WiSARD-based predictor with the state-of-the-art (TAGE-SC-L) and the Multiperspective Perceptron predictor. We labeled them WNN, T, and MP, respectively.

5.2 Sensitivity analysis

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

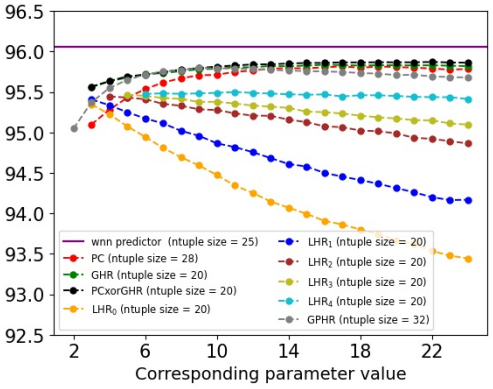
We show and explain the full results in B. In Figure 5.2, we report the best curves achieving the highest accuracy for each feature and dataset. In the legend, the n-tuple size used in each curve is indicated within parentheses for each case.

In datasets I1 and I2, the PC is the most relevant feature (Figures 5.2a and 5.2b) since the accuracy increases as its corresponding parameter also increases. On the

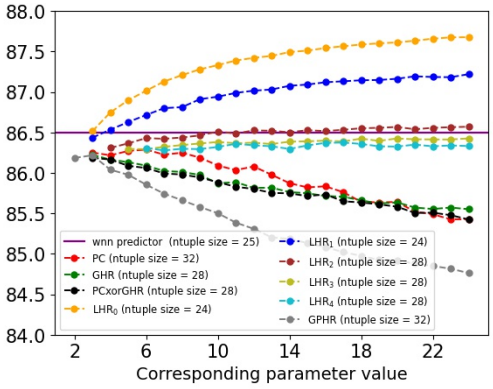


(a) I1

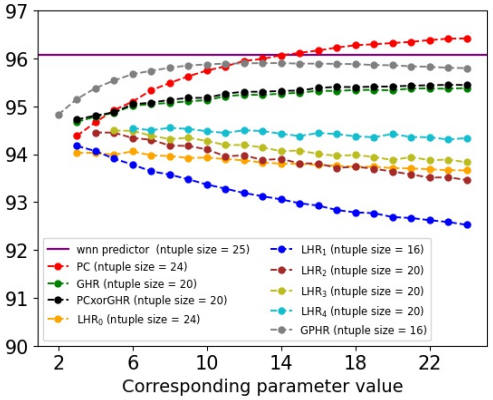
(b) I2



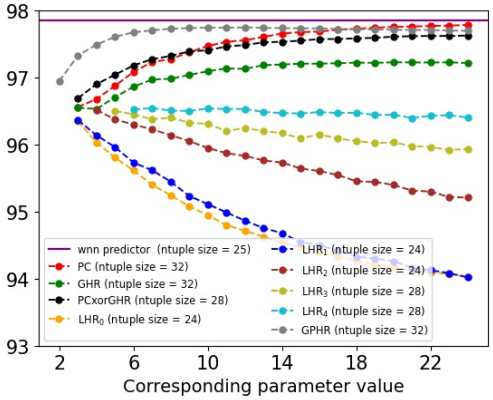
(c) M1



(d) M2



(e) S1



(f) S2

Figure 5.2: 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 Figures A.1 - A.6 from Appendix A.

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

Table 5.5: 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 5.4.

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.

A different situation occurs in datasets M1 and M2. In both datasets, the accuracy increases for features PCxorGHR and LHR_0 respectively (Figures 5.2c and 5.2d). 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 (Figures 5.2e and 5.2f). 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. This happens in the group integers (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 the first experiment (Table 5.4). Clearly, in datasets I1 and I2, there is at least one curve that surpasses the previous result for some value of the parameters (Figure 5.2a and 5.2b). In datasets M1 and S2, there is no such a curve that outperforms the horizontal line (Figure 5.2c and 5.2f). Lastly, in dataset M2, the results can be significantly more than 1% better than the previous WNN-predictor result (Figure 5.2d); while in dataset S1, this occurs for large values of the parameter associated with PC (Figure 5.2e).

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. Table 5.6 shows the results for best parameters configuration accordingly. We present the accuracy results obtained in this study in Table 5.7, and we consolidate them with the results of Table 5.4 in Table 5.8. In the experimental approach of this section, it is more important to outperform the accuracy described in Table 5.4 versus the necessity of smaller input sizes. This dissertation achieves this objective for all datasets compared to the previous WNN version, while the best proposed predictors only outperform the state-of-the-art in the datasets M2, S1, and S2. It is pertinent to emphasize the results in dataset M2, where the accuracy obtained is at least 2.3% higher than TAGE-SC-L and Multiperspective Perceptron counterparts.

Dataset	n-tuple	a	b	c	d_0	d_1	d_2	d_3	d_4	e	Input size
I1	17	24	2	2	2	8	3	4	5	1	992
I2	27	24	6	6	2	2	3	4	0	2	1127
M1	24	72	24	72	16	16	16	12	24	16	6044
M2	23	10	10	10	150	10	15	20	25	5	5200
S1	38	140	10	10	4	4	4	10	12	8	4678
S2	38	150	16	16	2	2	8	16	26	8	5274

Table 5.6: Best parameters configuration for each dataset

Dataset	Accuracy(Best)
I1	99.8067±.0018
I2	99.9786±.0009
M1	96.1322±.0157
M2	88.1783±.0202
S1	96.8521±.0179
S2	98.0946±.0110

Table 5.7: Accuracy results for the best parameters configuration (Table 5.6) for each dataset

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

Table 5.8: Comparison of the accuracy results reported in Tables 5.4 and 5.6

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

Table 5.9: Most relevant feature of the input for each dataset

When comparing the results from Table 5.6 to the sensitivity analysis (Figures A.1 - A.6), a direct correlation between the parameters and the features from the input it is observed, as expected. Table 5.9 summarizes this correlation. Manifestly, 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 related work [43]. Thus, this dissertation performed an additional analysis using the best predictors for each dataset, according to the results from Table 5.6.

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 (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.

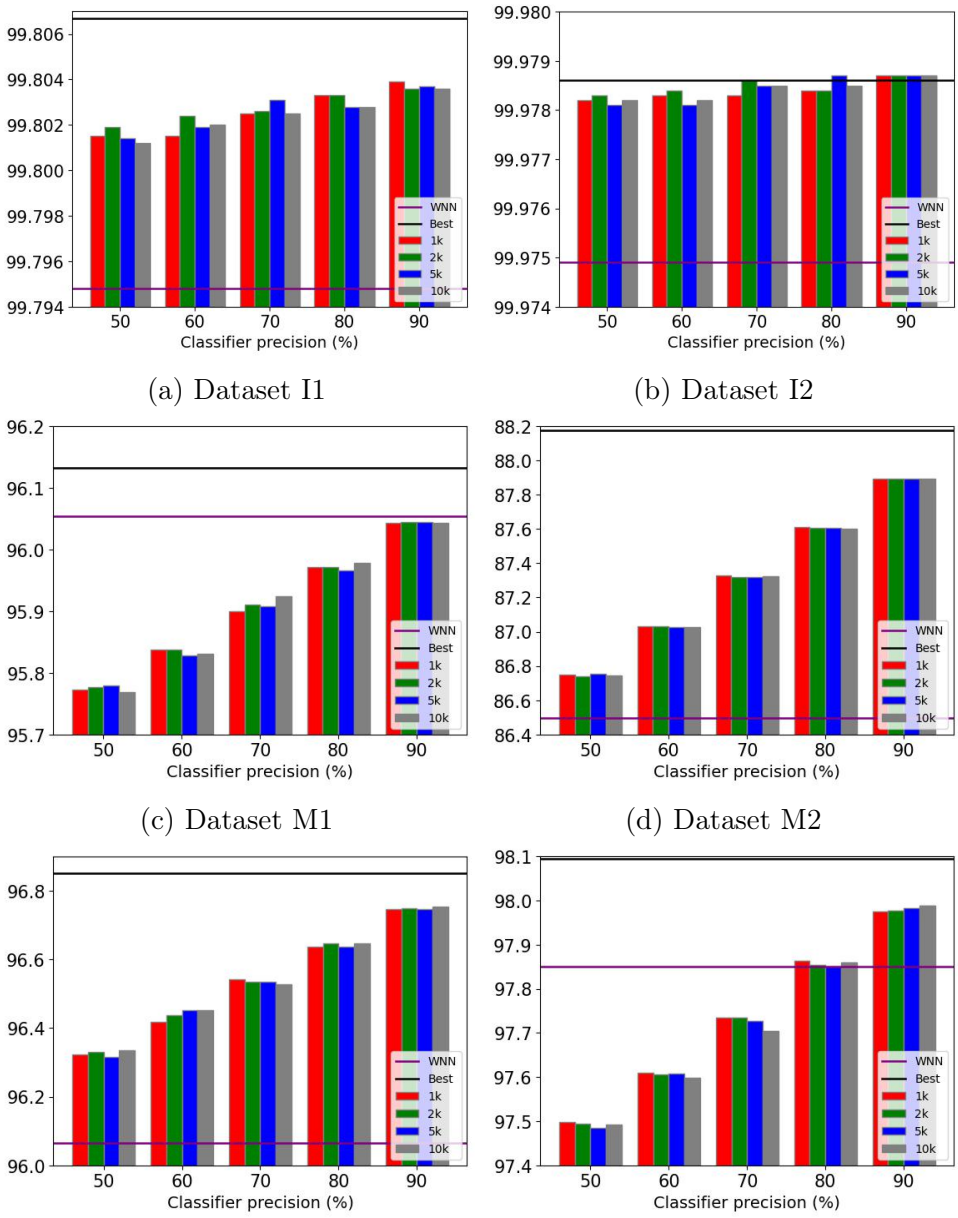
This dissertation implemented the classification algorithm of specialized predictors by defining the rate of precision of its selection. To quantify the final accuracy of the prediction, we divide the number of instructions in each dataset into blocks of a fixed size, where the prediction of each block of instructions where performed by all the specialized predictors. We summarize these results in Figure 5.3. The results of the specialized predictors and the initial version of the WiSARD-based predictor (Table 5.4), 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, (Figure 5.3a and Figure 5.3a). 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 (Figures 5.3a - 5.3f).



(a) Dataset I1

(b) Dataset I2

(c) Dataset M1

(d) Dataset M2

(e) Dataset S1

(f) Dataset S2

Figure 5.3: 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.

Chapter 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, it can be implemented in hardware and real-time applications. One interesting and potential area of application of WNN is computer architecture. Specifically, WNN can be explored as part of the conditional branch predictor architecture, a well-established technology implemented in nearly all modern computer processors.

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

First, through a pseudo-exhaustive hyperparameter search, this dissertation 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 fewer hardware resources), the proposed predictor predictor achieves, on average, similar accuracies to the TAGE-SC-L and outperforms the Multiperspective Perceptron by approximately 0.09%.

Next, this dissertation 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, the 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 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 [44] 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 one can interpret 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 this dissertation 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.

References

[1] PENNEY, D. D., CHEN, L. "A survey of machine learning applied to computer architecture design", *arXiv preprint arXiv:1909.12373*, 2019.

[2] JIMÉNEZ, D. A., LIN, C. "Dynamic branch prediction with perceptrons". In: *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pp. 197–206. IEEE, 2001.

[3] JIMÉNEZ, D. A. "Fast path-based neural branch prediction". In: *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pp. 243–252. IEEE, 2003.

[4] SMITH, A. "Branch prediction with neural networks: Hidden layers and recurrent connections", *Department of Computer Science University of California, San Diego La Jolla, CA*, v. 92307, 2004.

[5] TARSA, S. J., LIN, C.-K., KESKIN, G., et al. "Improving branch prediction by modeling global history with convolutional neural networks", *arXiv preprint arXiv:1906.09889*, 2019.

[6] MAO, Y., ZHOU, H., GUI, X., et al. "Exploring convolution neural network for branch prediction", *IEEE Access*, v. 8, pp. 152008–152016, 2020.

[7] MICHAUD, P. "An alternative tage-like conditional branch predictor", *ACM Transactions on Architecture and Code Optimization (TACO)*, v. 15, n. 3, pp. 1–23, 2018.

[8] ALEKSANDER, I., THOMAS, W., BOWDEN, P. "WISARD·a radical step forward in image recognition", *Sensor Review*, v. 4, n. 3, pp. 120–124, 1984. ISSN: 0260-2288. doi: 10.1108/eb007637.

[9] HARRIS, S. L., HARRIS, D. *Digital Design and Computer Architecture, RISC-V Edition*. USA, Morgan Kaufmann, 2021.

[10] HENNESSY, J. L., PATTERSON, D. A. *Computer architecture: A Quantitative Approach*. USA, Elsevier, 2019.

[11] PATTERSON, D. A., HENNESSY, J. L. *Computer organization and design RISC-V edition: the hardware/software interface*. USA, Morgan Kaufmann, Cambridge, 2017.

[12] JIANG, T., WU, N., ZHOU, F., et al. "Design of a High Performance Branch Predictor Based on Global History Considering Hardware Cost". In: *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, pp. 422–426. IEEE, 2021.

[13] KONFLANZ, D. M. "Investigating hierarchical temporal memory networks applied to dynamic branch prediction", 2019.

[14] MITTAL, S. "A survey of techniques for dynamic branch prediction", *Concurrency and Computation: Practice and Experience*, v. 31, n. 1, pp. e4666, 2019.

[15] LIN, C.-K., TARSA, S. J. "Branch prediction is not a solved problem: Measurements, opportunities, and future directions", *arXiv preprint arXiv:1906.08170*, 2019.

[16] SMITH, J. "A study of branch prediction techniques". In: *Proceedings of the 8th Annual Symposium on Computer Architecture*, pp. 135–147, 1981.

[17] SIPSER, M. *Introduction to the Theory of Computation*. Introduction to the Theory of Computation. USA, Cengage Learning, 2012.

[18] PAN, S.-T., SO, K., RAHMEH, J. T. "Improving the accuracy of dynamic branch prediction using branch correlation". In: *Proceedings of the fifth international conference on Architectural support for programming languages and operating systems*, pp. 76–84, 1992.

[19] YEH, T.-Y., PATT, Y. N. "Alternative implementations of two-level adaptive branch prediction", *ACM SIGARCH Computer Architecture News*, v. 20, n. 2, pp. 124–134, 1992.

[20] YEH, T.-Y., PATT, Y. N. "A comparison of dynamic branch predictors that use two levels of branch history". In: *Proceedings of the 20th annual international symposium on computer architecture*, pp. 257–266, 1993.

[21] MCFARLING, S. *Combining branch predictors*. Relatório técnico, Citeseer, 1993.

[22] JIMÉNEZ, D. A., LIN, C. "Perceptron learning for predicting the behavior of conditional branches". In: *IJCNN'01. International Joint Conference on*

Neural Networks. Proceedings (Cat. No. 01CH37222), v. 3, pp. 2122–2127. IEEE, 2001.

[23] JIMÉNEZ, D. A., LIN, C. "Neural methods for dynamic branch prediction", *ACM Transactions on Computer Systems (TOCS)*, v. 20, n. 4, pp. 369–397, 2002.

[24] LOH, G. H., JIMENEZ, D. A. "Reducing the power and complexity of path-based neural branch prediction". In: *Proceedings of the 5th Workshop on Complexity Effective Design (WCED5)*, pp. 1–8, 2005.

[25] JIMÉNEZ, D. A., LOH, G. H. "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)*, pp. 55–62. IEEE, 2006.

[26] AMANT, R. S., JIMÉNEZ, D. A., BURGER, D. "Low-power, high-performance analog neural branch prediction". In: *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pp. 447–458. IEEE, 2008.

[27] JIMÉNEZ, D. A. "Piecewise linear branch prediction". In: *32nd International Symposium on Computer Architecture (ISCA'05)*, pp. 382–393. IEEE, 2005.

[28] JIMÉNEZ, D. A. "Generalizing neural branch prediction", *ACM Transactions on Architecture and Code Optimization (TACO)*, v. 5, n. 4, pp. 1–27, 2009.

[29] JIMÉNEZ, D. A. "Multiperspective perceptron predictor". In: *5th JILP Workshop on Computer Architecture Competitions: Championship Branch Prediction (CBP-5)*, p. 5, 2016.

[30] GRAYSON, B., RUPLEY, J., ZURASKI, G. Z., et al. "Evolution of the samsung exynos cpu microarchitecture". In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 40–51. IEEE, 2020.

[31] SEZNEC, A., MICHAUD, P. "A case for (partially) TAgged GEometric history length branch prediction", *The Journal of Instruction-Level Parallelism*, v. 8, pp. 23, 2006.

[32] SEZNEC, A. "Tage-sc-l branch predictors". In: *JILP-Championship Branch Prediction*, p. 9, 2014.

[33] MAO, Y., HUIYANG, Z., GUI, X. "Exploring deep neural networks for branch prediction", *ECE Department, NC University*, 2017.

[34] BLEDSOE, W. W., BROWNING, I. "Pattern recognition and reading by machine". In: *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, pp. 225–232, 1959.

[35] MCCULLOCH, W. S., PITTS, W. "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, v. 5, n. 4, pp. 115–133, 1943.

[36] ALEKSANDER, I., DE GREGORIO, M., FRANÇA, F. M. G., et al. "A brief introduction to weightless neural systems." In: *ESANN*, pp. 299–305. Citeseer, 2009.

[37] SPRUSTON, N., STUART, G., HÄUSSER, M. "Dendritic integration", *Dendrites*, pp. 231–271, 1999.

[38] ALEKSANDER, I. "Ideal neurons for neural computers", *Parallel Processing in Neural Systems and Computers*, pp. 225–228, 1990.

[39] FILHO, L. A. L., OLIVEIRA, L. F., FILHO, A. L., et al. "Prediction of Palm Oil Production with an Enhanced n-Tuple Regression Network". In: *ESANN*, p. 6, 2019.

[40] GRIECO, B. P., LIMA, P. M., DE GREGORIO, M., et al. "Producing pattern examples from "mental" images", *Neurocomputing*, v. 73, n. 7-9, pp. 1057–1064, 2010.

[41] JIMÉNEZ, D. "Idealized piecewise linear branch prediction", *Journal of Instruction-Level Parallelism*, v. 7, pp. 1–11, 2005.

[42] SHKADAREVICH, D. "Branch Prediction". `https://www.kaggle.com/dmitryshkadarevich/branch-prediction`, 2020.

[43] KHAN, T. A., UGUR, M., NATHELLA, K., et al. "Whisper: Profile-guided branch misprediction elimination for data center applications". In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 19–34. IEEE, 2022.

[44] SANTIAGO, L., VERONA, L., RANGEL, F., et al. "Weightless neural networks as memory segmented bloom filters", *Neurocomputing*, v. 416, pp. 292–304, 2020.

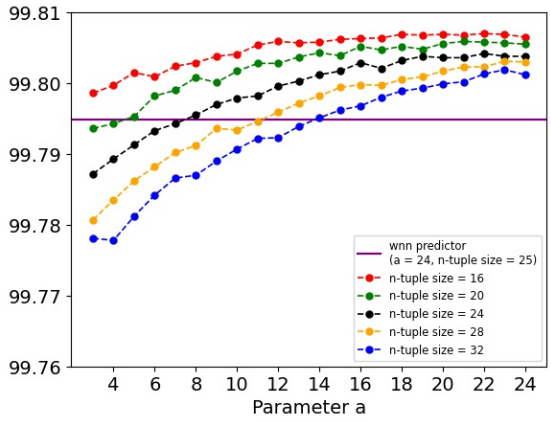
Appendix A

Supplementary sensitivity analysis

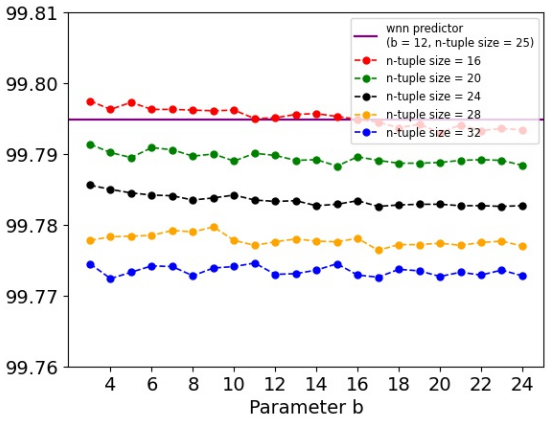
This appendix shows the full results of the sensitivity analysis exhibited in Figures A.1 - A.6. In all subfigures, the vertical and horizontal axis represent, respectively, the accuracy and the parameter associated to each feature according to relation 3.1. Each dashed curve represent the behavior for a particular n-tuple size. In addition, the continuous horizontal line exhibits the previous experimental result (Table 5.4). For each feature, this horizontal line also references, in parentheses, the corresponding parameter value and the n-tuple size previously employed.

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, 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 [36].

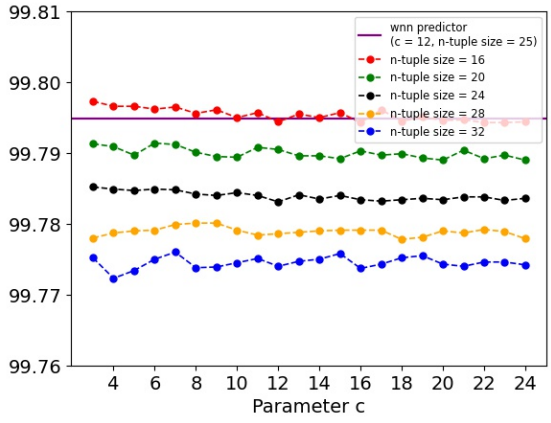
In addition, in some cases we observe oscillatory trends of the accuracy curves, especially in the results of datasets I1 and I2 (Figures A.1 and A.2). This is explained by the fact that binary input size is not a multiple of the n-tuple size in these cases. The WiSARD-based predictor is designed to 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.



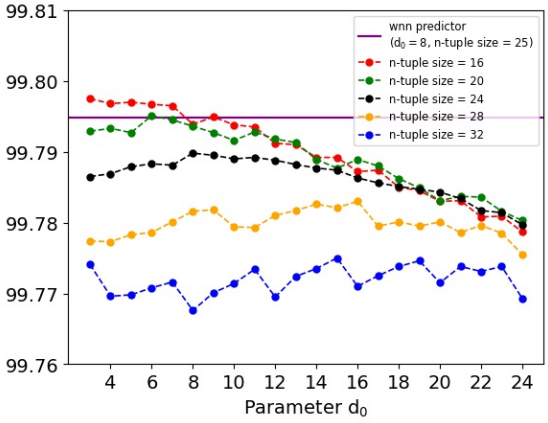
(a) PC



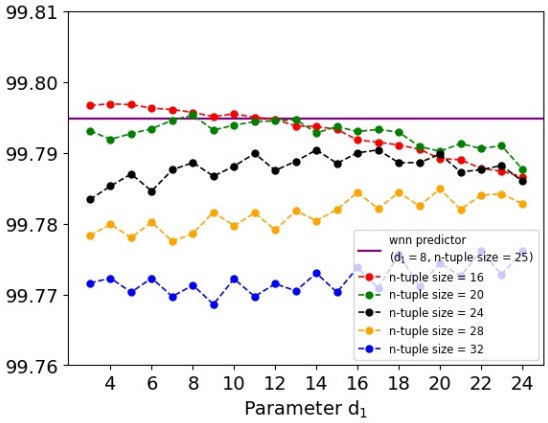
(b) GHR



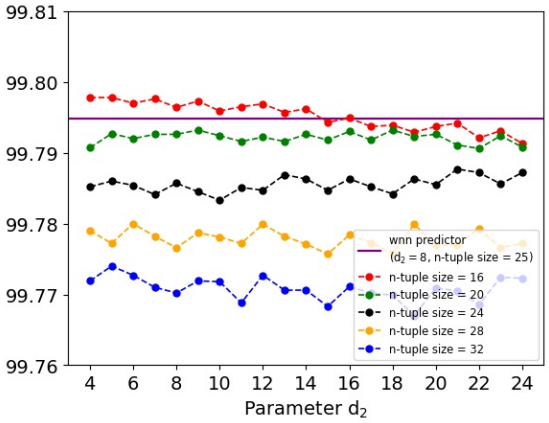
(c) PCxorGHR



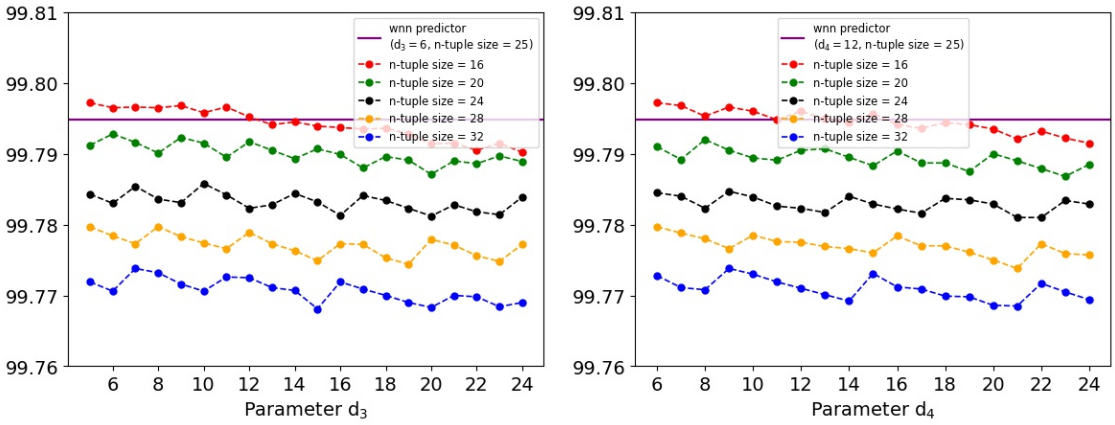
(d) LHR_0



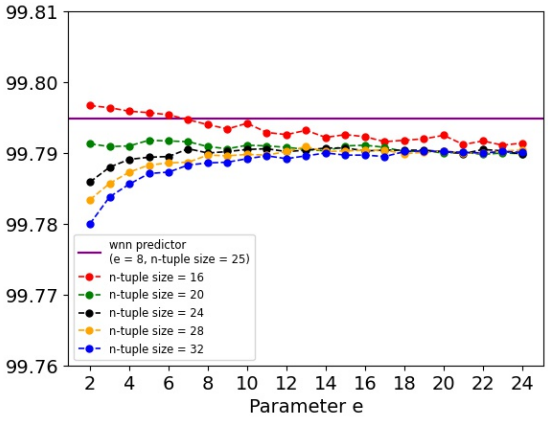
(e) LHR_1



(f) LHR_2

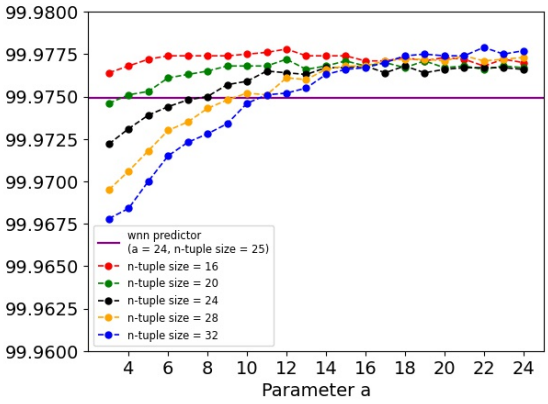


(g) LHR_3 (h) LHR_4

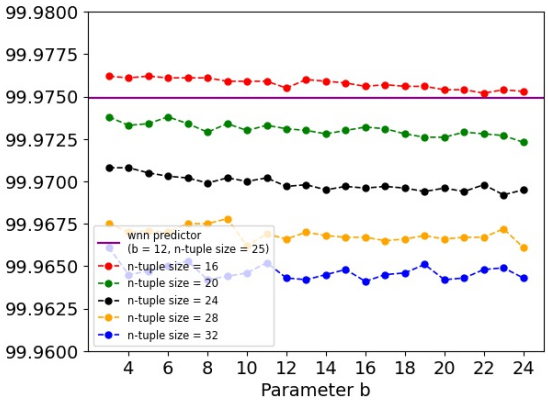


(i) GPHR

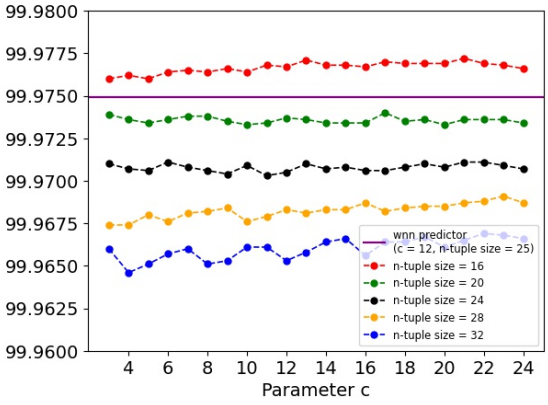
Figure A.1: Sensitivity Analysis for dataset I1. The PC is the most important feature (Figure A.1a) since the accuracy increases as its corresponding parameter a increases when compared to the other features (Figures A.1b - A.1i). For the other features, the accuracy drops or remains oscillating (Figures A.1b - A.1h). 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 (Figure A.1i). In addition, in nearly all cases, the accuracy drops as n-tuple size increases.



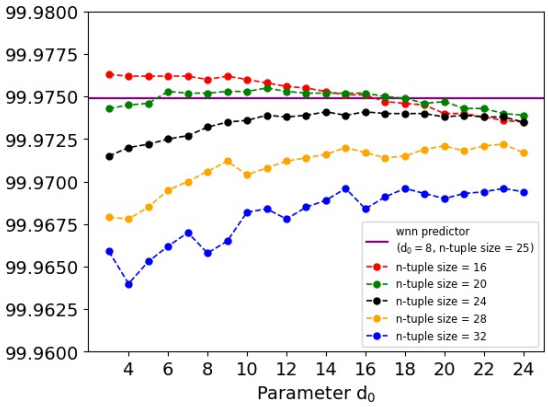
(a) PC



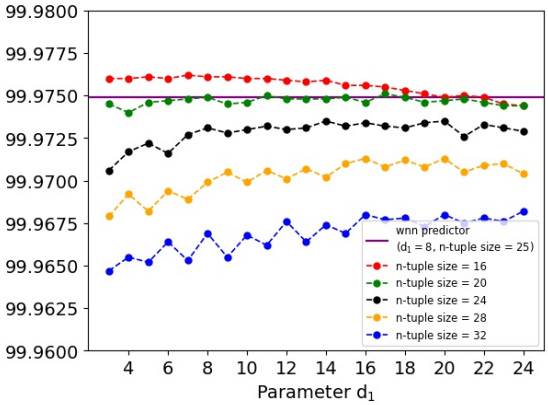
(b) GHR



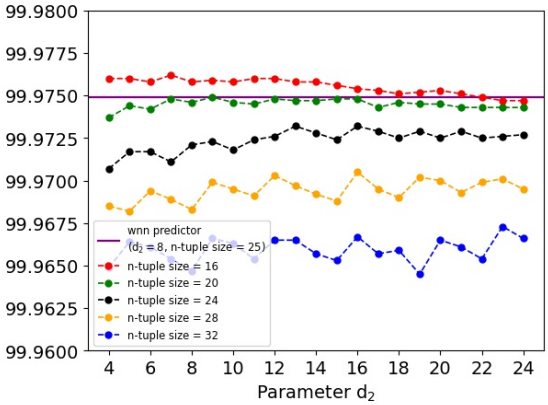
(c) PCxorGHR



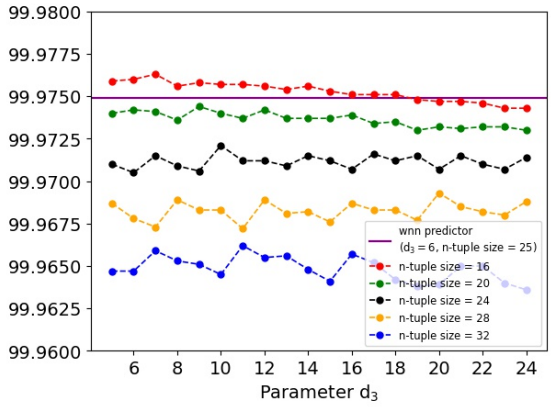
(d) LHR_0



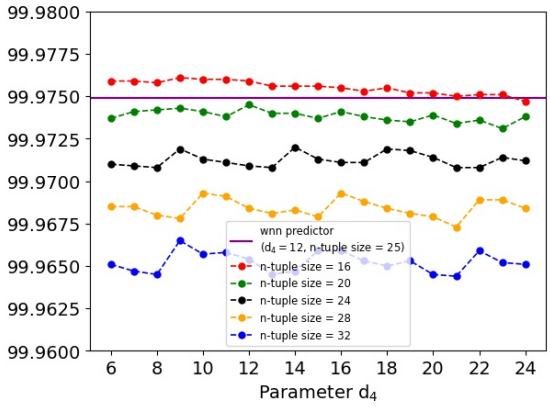
(e) LHR_1



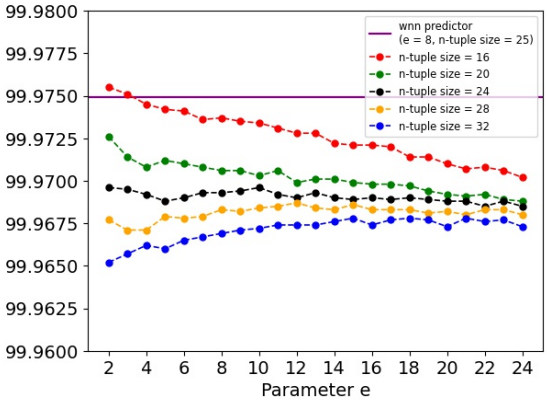
(f) LHR_2



(g) LHR_3

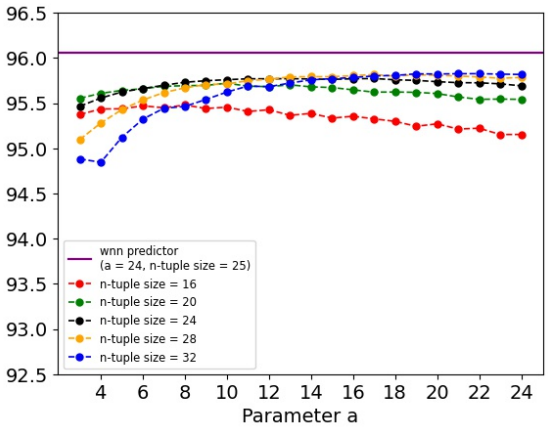


(h) LHR_4

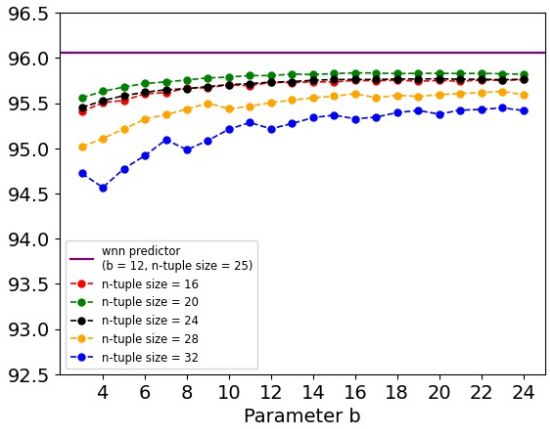


(i) GPHR

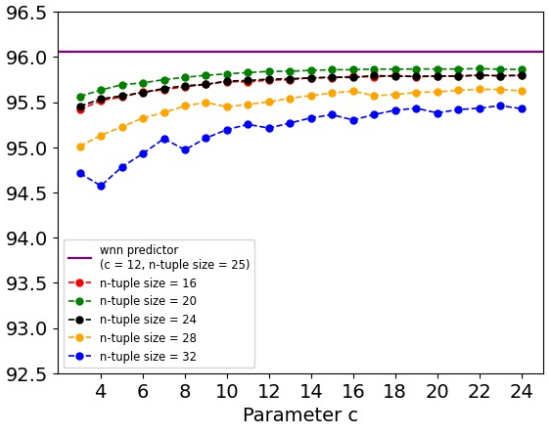
Figure A.2: Sensitivity Analysis for dataset I2. In this case the PC and the PCxorGHR are the most relevant features since the accuracy increases as its corresponding parameter a and c increases when compared to the other features (Figures A.2a and A.2c). In nearly all cases (Figures A.2b - A.2i) the 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 (Figures A.2f - A.2h).



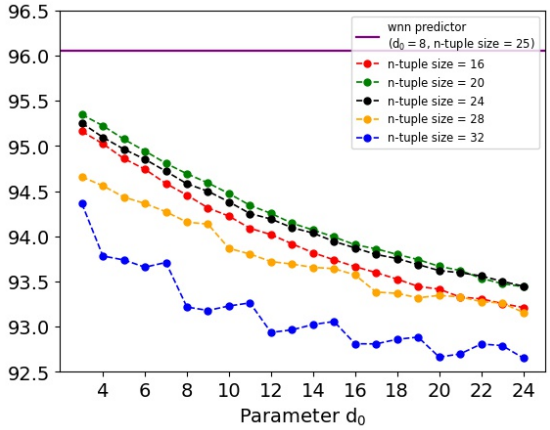
(a) PC



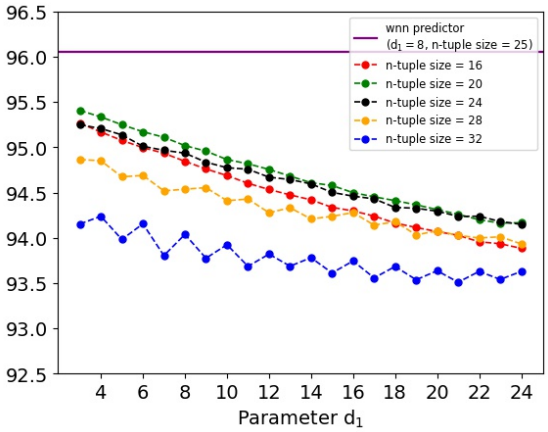
(b) GHR



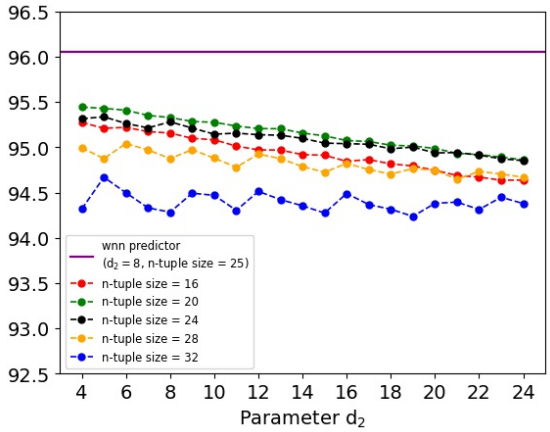
(c) PCxorGHR



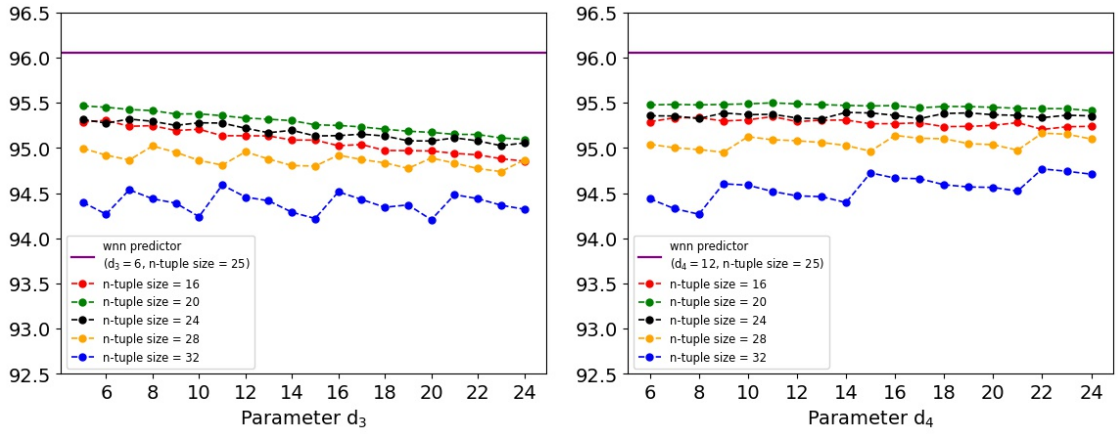
(d) LHR_0



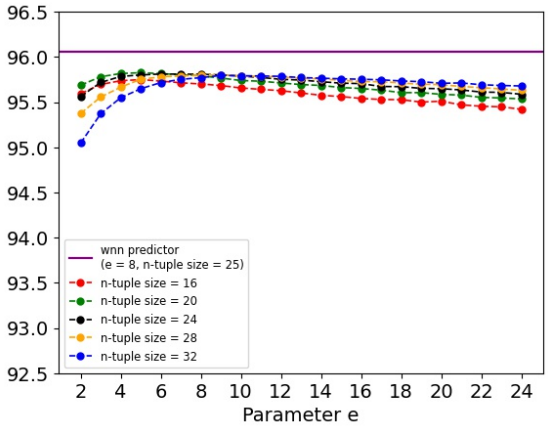
(e) LHR_1



(f) LHR_2

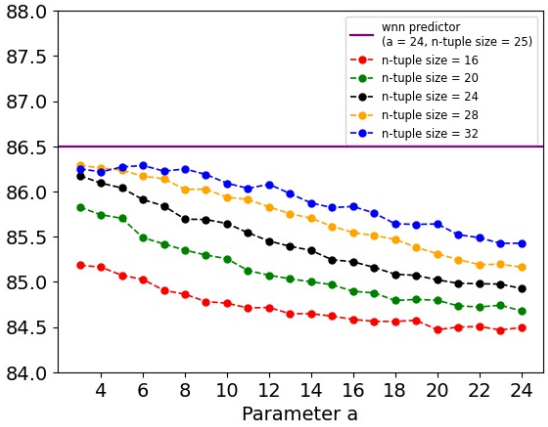


(g) LHR_3 (h) LHR_4

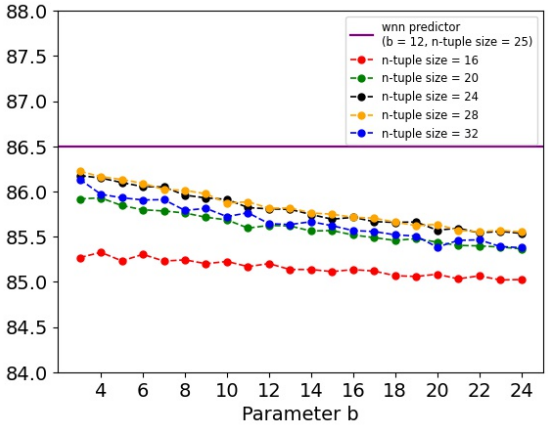


(i) GPHR

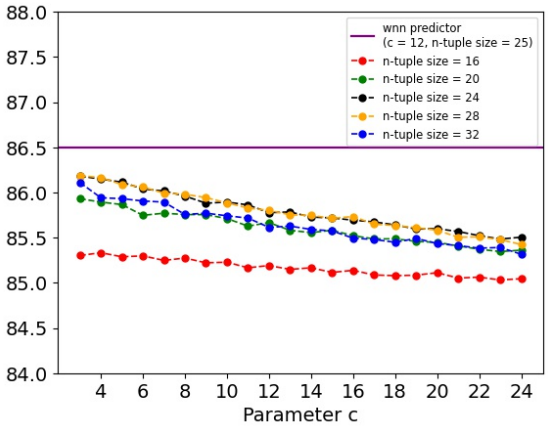
Figure A.3: Sensitivity Analysis for dataset M1. The results show differences in accuracy trends for each feature in each case. The accuracy increases or decreases smoothly for the features PC, GHR and PCxorGHR (Figures A.3a - A.3c), but substantially drops for LHR_0, LHR_1, LHR_2 (Figures A.3d - A.3f). Interestingly, the accuracy initially increases and then drops smoothly for large values of e for GPHR (Figure A.3i). In most cases, the accuracy decreases for large n-tuple sizes (Figures A.3b - A.3h).



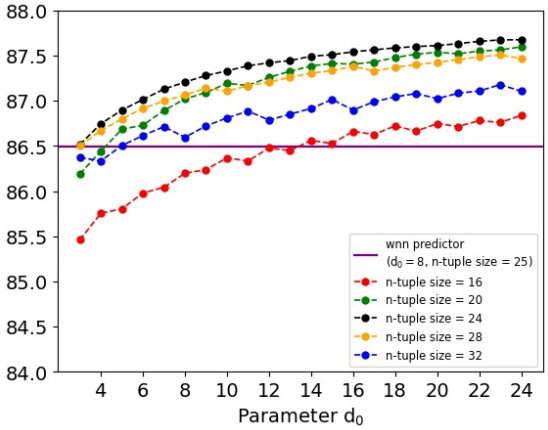
(a) PC



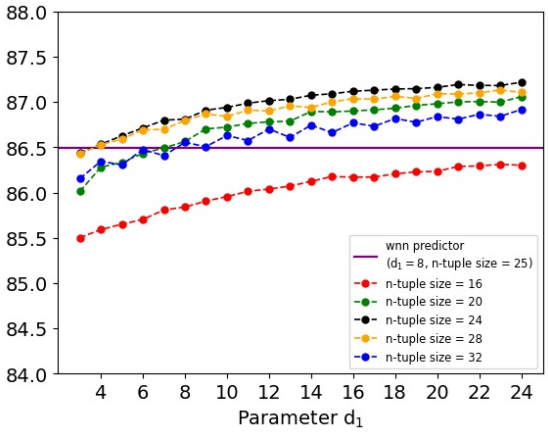
(b) GHR



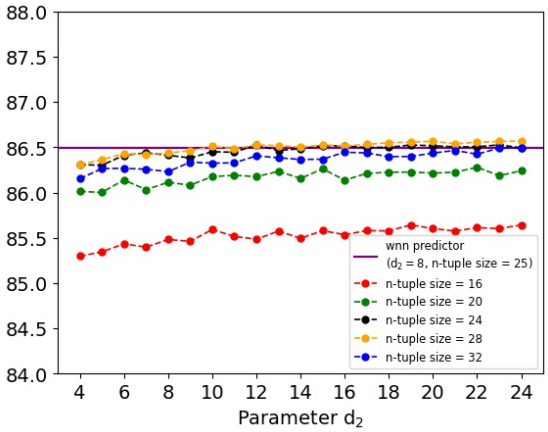
(c) PCxorGHR



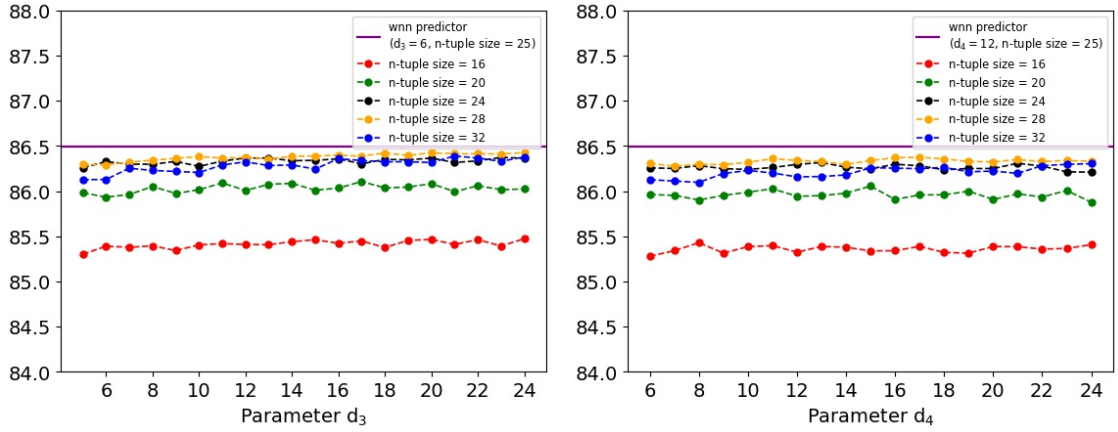
(d) LHR_0



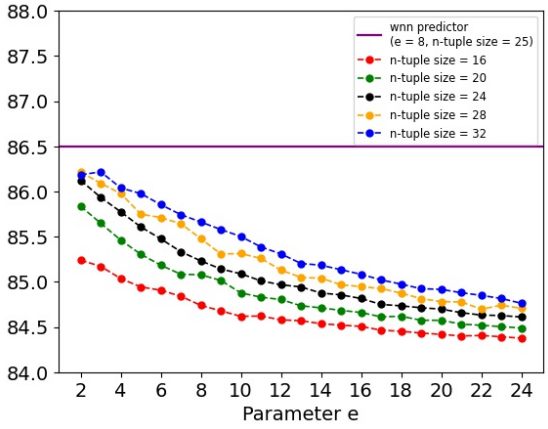
(e) LHR_1



(f) LHR_2

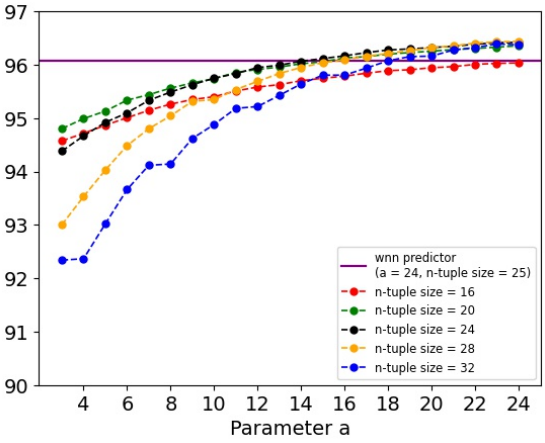


(g) LHR_3 (h) LHR_4

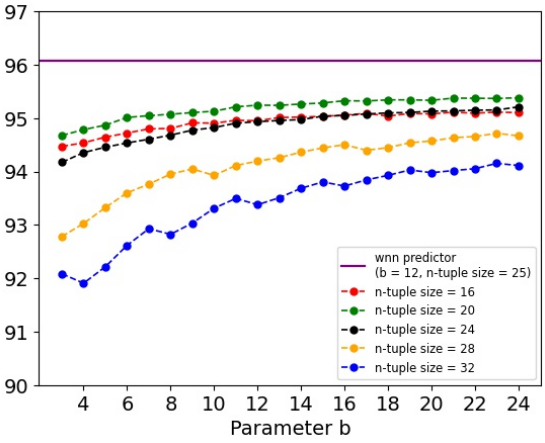


(i) GPHR

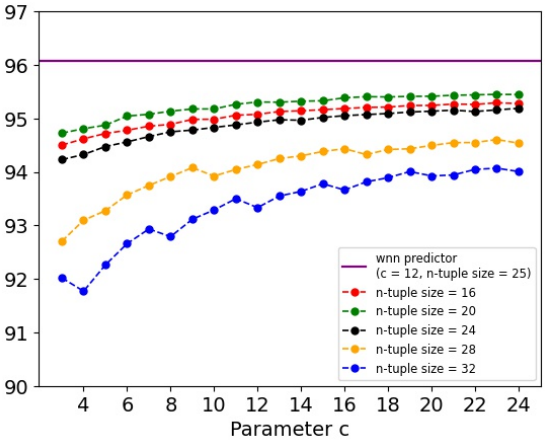
Figure A.4: Sensitivity Analysis for dataset M2. In this case, the accuracy increases for LHR_0, LHR_1, LHR_2 (Figures A.4d - A.4f); remains nearly constant for the features LHR_3 and LHR_4 (Figures A.4g and A.4h) and drops significantly for the other features (Figures A.4a, A.4b, A.4c, A.4i). For this dataset it is important to highlight the feature GPHR, since the accuracy degrades significantly as the parameter e increases (Figure A.4i). In addition, in almost all cases the accuracy increases as the n-tuple size increases.



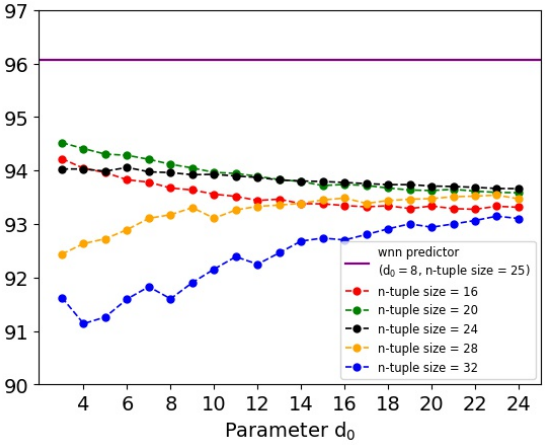
(a) PC



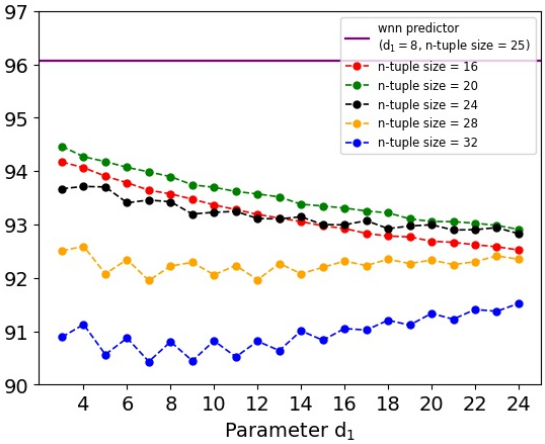
(b) GHR



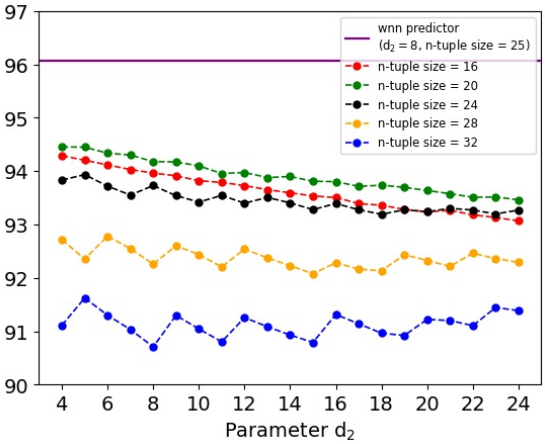
(c) PCxorGHR



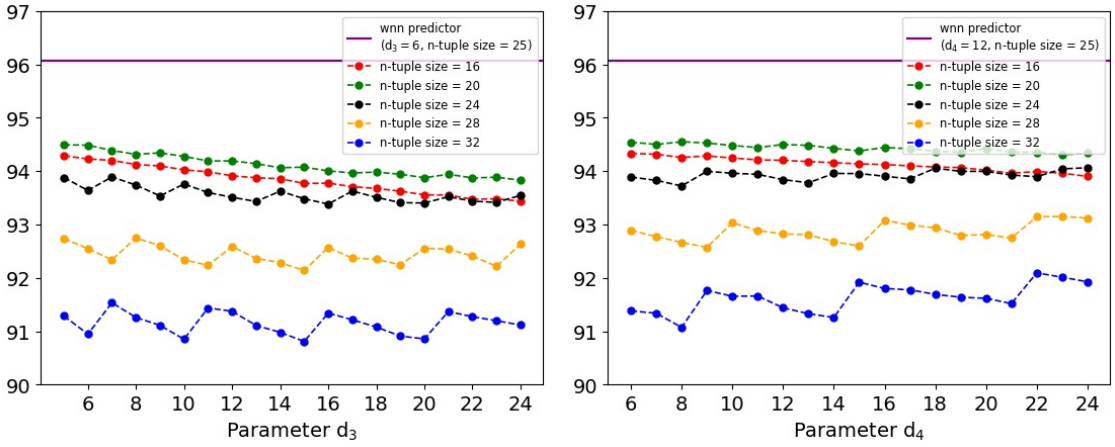
(d) LHR_0



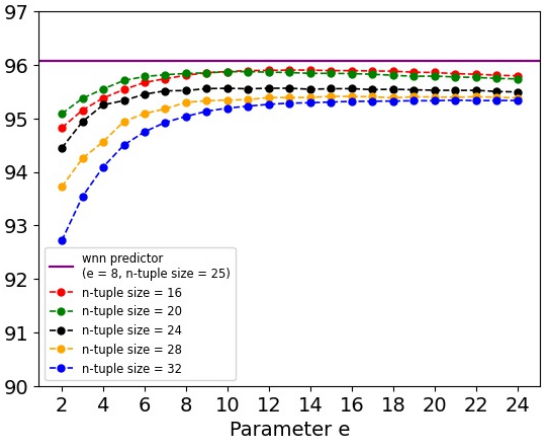
(e) LHR_1



(f) LHR_2

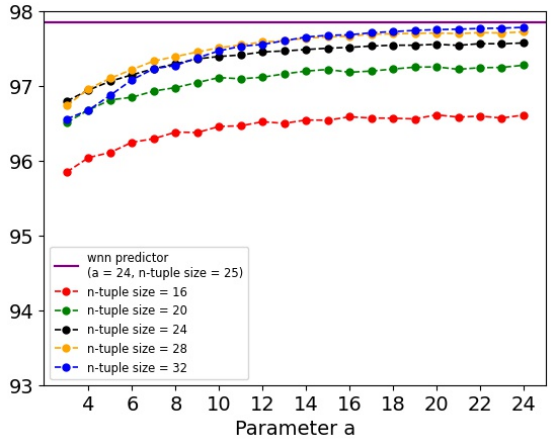


(g) LHR_3 (h) LHR_4

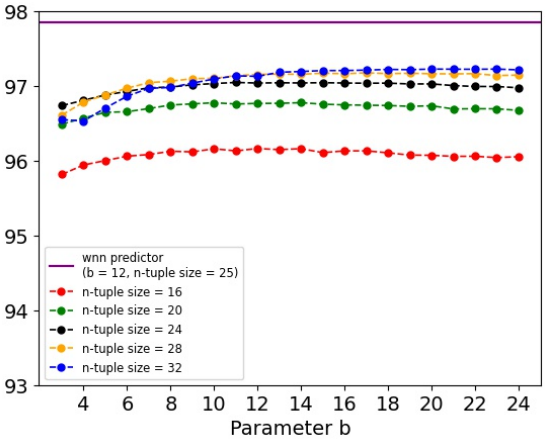


(i) GPHR

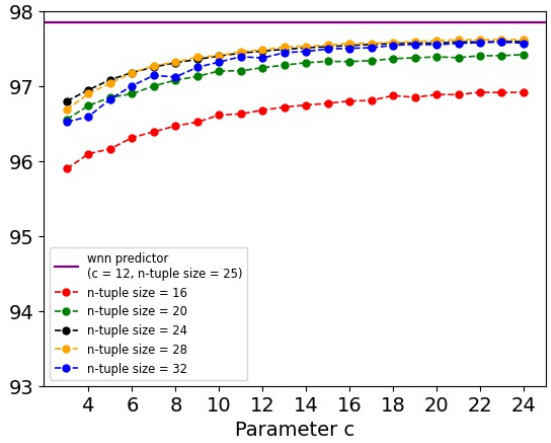
Figure A.5: Sensitivity Analysis for dataset S1. The most relevant features are the PC and GPHR (Figures A.5a and A.5i) since accuracy achieves the highest values, particularly for high values of the parameter a. The accuracy also increases for the features GHR and PCxorGHR (Figures A.5b and A.5c) but decreases for all other LHR type features (Figures A.5d - A.5h). Furthermore, in almost all cases the best and worst curves correspond to n-tuple sizes 20 and 32 respectively.



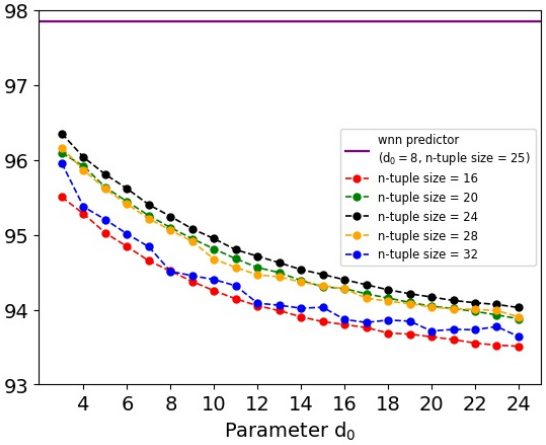
(a) PC



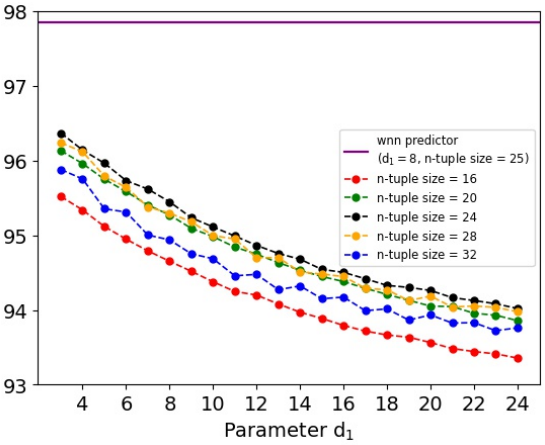
(b) GHR



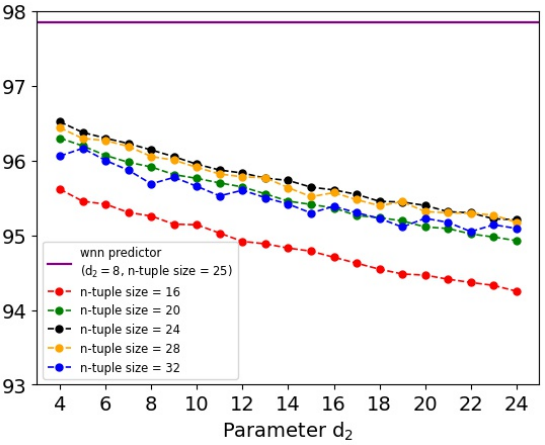
(c) PCxorGHR



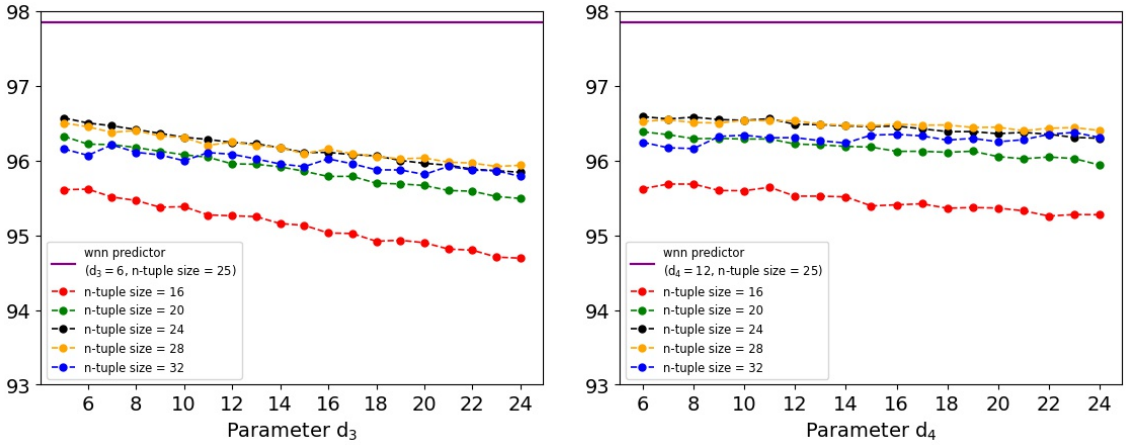
(d) LHR_0



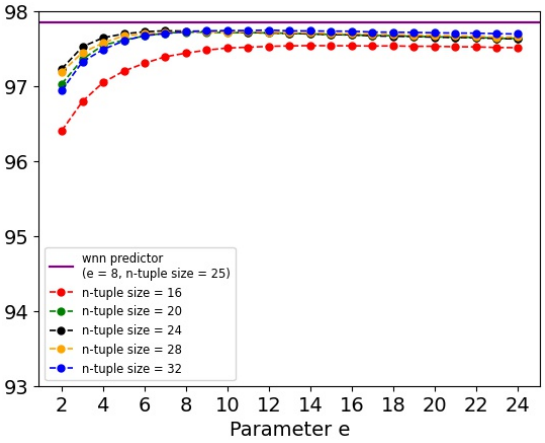
(e) LHR_1



(f) LHR_2



(g) LHR_3 (h) LHR_4



(i) GPHR

Figure A.6: Sensitivity Analysis for dataset S2. The most important features are the PC and GPHR (Figures A.6a and A.6i) since the accuracy achieves the highest values when compared to the other features. The accuracy also increases for the features GHR and PCxorGHR (Figures A.6b and A.6c) but decreases significantly for all other LHR type features (Figures A.6d - A.6h). 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.

Appendix B

Published papers

This appendix includes the first pages of works developed in authorship or co-authorship during the preparation of the current dissertation.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

A WiSARD-based conditional branch predictor

Luis A. Q. Villon[1], Zachary Susskind[2], Alan T. L. Bacellar[1], Igor D. S. Miranda[3], Leandro S. de Araújo[4], Priscila M. V. Lima[1], Mauricio Breternitz Jr.[5], Lizy K. John[2], Felipe M. G. França[1,6] and Diego L. C. Dutra[1] *

1- UFRJ, Rio de Janeiro, Brazil, 2- UT Austin, Austin, USA,
3- UFRB, Cruz das Almas, Brazil, 4- UFF, Niterói, Brazil,
5- ISCTE, Lisbon, Portugal, 6- Instituto de Telecomunicações, Portugal

Abstract. Conditional branch prediction is a technique used to speculatively execute instructions before knowing the direction of conditional statements. Perceptron-based predictors have been extensively studied, however, they need large input sizes for the data to be linearly separable. To learn nonlinear functions from the inputs, we propose a conditional branch predictor based on the WiSARD weightless neural network model and compare it with two state-of-the-art predictors: TAGE-SC-L and the Multiperspective Perceptron. We show that the WiSARD-based 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.

1 Introduction

In recent decades, different types of neural networks have been applied to address several topics in computer microarchitecture [1]. Specifically, innovative techniques for implementing branch prediction were covered using perceptron [2] [3], feedforward neural networks [4], recurrent networks and convolutional networks [5] [6].

Weightless neural networks (WNNs) are a category of neural model 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. The main advantage of WNNs is the capacity 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) [7] is considered to be the first WNN to achieve a 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, the WiSARD model is an attractive alternative to traditional neural-based predictors. Nevertheless, there is no previous work, to our knowledge, that uses WiSARD in a conditional branch predictor. Consequently, this work aims to explore the potential gain of accuracy using a WiSARD-based branch predictor that requires smaller input sizes when compared to state-of-the-art predictors, the TAGE-SC-L and the Multiperspective perceptron. Our results show

*This paper was partially supported by (a) CAPES - Brazil - Finance Code 001; (b) CNPq - Brazil; (c) Fundação para a Ciência e a Tecnologia, I.P. (FCT): ISTAR Projects: UIDB/04466/2020 and UIDP/04466/2020, and; (d) FCT/COMPETE/FEDER, FCT/CMU IT Project FLOYD: POCI-01-0247-FEDER-045912.