Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
COPPE UFRJ

EXTENDING WISARD TO PERFORM ENSEMBLE LEARNING, REGRESSION, MULTI-LABEL, AND MULTI-MODAL TASKS

Leopoldo André Dutra Lusquino Filho

Tese de Doutorado 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 Doutor em Engenharia de Sistemas e Computação.

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Rio de Janeiro
Julho de 2021

EXTENDING WISARD TO PERFORM ENSEMBLE LEARNING, REGRESSION, MULTI-LABEL, AND MULTI-MODAL TASKS

Leopoldo André Dutra Lusquino Filho

TESE 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 DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Aprovada por: Priscila Machado Vieira Lima Ph.D.
Felipe Maia Galvão França Ph.D.
Diego Leonel Cadette Dutra D.Sc.
Aluizio Fausto Ribeiro Araujo D.Phil.
David Alexis Owen Gamez Ph.D.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2021

Lusquino Filho, Leopoldo André Dutra

Extending WiSARD to Perform Ensemble Learning, Regression, Multi-label, and Multi-modal Tasks/Leopoldo André Dutra Lusquino Filho. – Rio de Janeiro: UFRJ/COPPE, 2021.

XXVII, 215 p.: il.; 29, 7cm.

Orientadores: Priscila Machado Vieira Lima

Felipe Maia Galvão França

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2021.

Referências Bibliográficas: p. 117 – 139.

1. WiSARD. 2. Weightless artificial neural network. 3. Cognitive architectures. 4. Multi-label classification. 5. Ensemble learning. 6. Non-parametric regression. 7. Multi-modal learning. I. Lima, Priscila Machado Vieira *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A Krishna-Balarama

Agradecimentos

nama om vishnu-padaya krishna-presthaya bhu-tale
srimate chandramukha swamin iti namine

nama om vishnu-padaya krishna-presthaya bhu-tale
srimate bhaktivedanta swamin iti namine

jaya sri krishna chaitanya prabhu nityananda
sri advaita gadadhara srivasadi gaura bhakta vrinda

hare krishna hare krishna krishna krishna hare hare
hare rama hare rama rama rama hare hare

Agradeço primeiramente a Krishna, o tesouro negro dos bosques de Vrindavana, e seu irmão Balarama, suavizante como o luar primaveril.

Agradeço a Srila Prabhupada, por distribuir tão misericordiosamente o mais valioso néctar indiscriminadamente, e ao seu mais querido servo, Chandramukha Swami, por ter me aceitado como seu discípulo e ressignificado toda minha vida.

Agradeço aos meus pais, Carmen e Balabhadra, por terem me estimulado a sonhar o tanto quanto eu pudesse, por terem me dado um lar tranquilo e feliz e por terem me ensinado o valor da retidão. Agradeço a minha melhor metade, Damodara, por me completar de tantas formas e ser a melhor companhia que eu poderia ter nesta misteriosa jornada da vida. Agradeço ao meu amado Krishna-Balarama, que me revigora completamente com seu doce sorriso. *krishna shaktir astu te*

Agradeço aos meus avós por toda sua generosidade, ao meu primo Raphael, por ter sido sempre uma inspiração, sua esposa Nina, uma irmã tardia, e sua filha Marina, minha querida afilhada.

Agradeço a ISKCON, que se tornou meu verdadeiro refúgio, especialmente ao Dhira Chaitanya, Bhagavan, Nityananda Raya e Chaitanya Chandrodaya, por todas as aventuras que passamos juntos. Ainda vamos salvar o mundo um dia!

Agradeço aos meus orientadores, professores Priscila e Felipe, que foram muito mais do que meus tutores acadêmicos durante estes anos. Muito obrigado pela amizade e confiança que me destinaram e que possamos continuar nossa parceria nos anos vindouros.

Agradeço aos meus destemidos irmãos de armas da Sociedade da RAM: Luiz, Gabriel e Aluizio. Certamente este trabalho não teria sido possível sem a colaboração de vocês. *elen sîla lúmenn' omentielvo*

Agradeço aos amigos dos laboratórios LAM, LabZero e LabIA, especialmente ao Hugo, Brunno, Victor, Santiago, João Victor e Eduardo. Com vocês, todo o fardo desta pesquisa se tornou mais leve e agradável.

Agradeço ao Vladimir e a NGD Systems por terem apoiado minha pesquisa e me propiciado oportunidades únicas de crescimento pessoal e profissional.

Agradeço a toda família do PESC/COPPE/UFRJ, por fazerem deste programa um ambiente tão amistoso e fértil academicamente. Agradeço também as agências de fomento CAPES, CNPq e COPPETEC, cuja ajuda foi fundamental durante minha pesquisa.

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

ESTENDENDO WISARD PARA APRENDIZAGEM DE COMITÊ E TAREFAS DE REGRESSÃO, MULTI-RÓTULO E MULTI-MODO

Leopoldo André Dutra Lusquino Filho

Julho/2021

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

Wilkie, Stonham and Aleksander's Recognition Device (WiSARD) é um modelo de aprendizado de máquina que não necessita de nenhum tipo de técnica de minimização de erro para aprender padrões. Este modelo utiliza RAMs como neurônios, sendo necessário apenas um processo de escrita em memória na sua fase de treinamento e leitura em memória na sua fase de classificação. WiSARD é uma rede neural sem peso, um tipo de modelo que tem sido usado exitosamente em pesquisas sobre arquiteturas cognitivas e consciência artificial. Esta tese propõe várias extensões para este modelo de forma a criar os componentes necessários para uma futura arquitetura cognitiva direcionada a emoções baseada em WiSARD. As contribuições deste trabalho incluem dois sistemas de classificação multi-rótulo utilizando WiSARD, cinco novos tipos de comitês utilizando tanto WiSARD, quanto sua extensão ClusWiSARD, dois modelos de regressão não-paramétrica e um de regressão logística baseados em WiSARD e um sistema multi-modal de predição de empatia baseado sem peso. Esta tese utiliza uma implementação da WiSARD baseada em tabelas de dispersão, instanciando apenas as posições de memória que foram de fato treinadas. Todos os modelos e sistemas criados para esta tese foram comparados com o estado-da-arte, sendo competitivos em alguns casos, enquanto preservam todas as qualidades da WiSARD canônica.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EXTENDING WISARD TO PERFORM ENSEMBLE LEARNING, REGRESSION, MULTI-LABEL, AND MULTI-MODAL TASKS

Leopoldo André Dutra Lusquino Filho

July/2021

Advisors: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

Wilkie, Stonham and Aleksander's Recognition Device (WiSARD) is a machine learning model that does not require any kind of error minimization technique to learn patterns. This model uses RAMs as neurons, requiring only one process of writing in memory in its training phase and reading in memory in its classification phase. WiSARD is a weightless artificial neural network, a kind of model that has been successfully used in cognitive architectures and artificial consciousness research. This thesis proposes several extensions for this model in order to create the necessary components for a future WiSARD-based emotion-drive cognitive architecture, The contributions of this work include two WiSARD-based multi-label classification systems, five new types of ensembles using both WiSARD, as well as its ClusWiSARD extension, two WiSARD-based non-parametric regression models, one WiSARD-based logistic regression model, and a weightless multi-modal empathy prediction system. This thesis uses a map-based WiSARD implementation, instantiating only the memory locations that were actually trained. All models and systems created for this thesis have been compared with state-of-the-art, being competitive in some cases, while preserving all the qualities of the canonical WiSARD.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wilkie, **S**tonham and **A**leksander **R**ecognition **D**evice (WiSARD)[6] is an artificial neural network that does not use weighted synapses to learn patterns. On the other hand, it possesses RAM(random-access-memory)-based neurons that stores the learnerd patterns. In a WiSARD model the learning of a pattern simply corresponds to writing in memory, whereas classification essentially corresponds to the reading of certain memory positions.

WiSARD belongs to a type of machine learning model known as Weightless Artificial Neural Networks (WANNs)[7] derived from the n-tuple Classifier[1]. The goal of this work is to extend the WiSARD model in order to create components for cognitive architectures from such extensions. Preliminary explorations of weightless models in such architectures have proven successful[8–16].

One of the advantages of using weightless models in the construction of cognitive architectures is that it is possible to build from the same type of neural network upper and lower layers analogous to those of the brain, since in this model the learning and classification parts could be built from the same components. Since recent studies[17–20] on the predictive brain have suggested that higher layers of the brain have a strong influence on the information processed in the lower layers, such modeling would preserve fidelity with the human brain. On the other hand, cognitive architectures that use deep models[21] tend to separate them from the cognitive components..

1.1 Motivation

Cognitive architectures were first proposed by Allan Newell[22], one of the pioneers of AI. He tried to establish a unified theory of cognition and defined the mind as a single system[23]. Its definition of cognition postulated that it was composed of the capacities of problem-solving, decision making, routine action, memory, learning, skill, perception, motor behavior, language, motivation, emotion, imagining, dreaming, and daydreaming. According to this unified theory, any plausible explanation of the mind should explain it as being achievable as a neural network[24].

Vernon et al.[25] define cognition as what "allows the system to act effectively, to adapt,and to improve". Panella et al.[21] connects cognition with an agent's sense of survival, in a way that a cognitive architecture will allow a system to adapt to survive in diverse environments. To do so, the architecture should provide the agent with the following capabilities: (i) sense the environment and themselves; (ii) create an inner representation of what they sense; (iii) be able to reason and make inferences about the environment; (iv) react to the environment; (v) learn and update knowledge; (vi) re-plan their course of action; (vii) actuate the new plan. Cognitive architecture surveys were provided by [26–29].

The concept of a theory for artificial consciousness-based only on neural machines was first presented by one of the WiSARD's creators, Professor Igor Aleksander, in [8]. Soon after he formulates the first theory of artificial consciousness based on the personal construct theory[30], and part of the premise that anticipation and prediction capabilities are the main drivers of mind. Aleksander then formulates his Fundamental Postulate of Consciousness, who tried to capture consciousness in a neural state machine, with sensory neurons and iconic transfer. This is better elaborated in Appendix F.

Aleksander also derives twelve corollaries from the Fundamental Postulate: (i) the brain is a state machine; (ii) inner neuron partitioning; (iii)conscious and unconscious states; (iv) perceptual learning and memory; (v) prediction; (vi) awareness of self; (vii) representation of meaning; (viii) learning utterances; (ix) learning language; (x) will; (xi) instinct; and (xii) emotion. According to this theory an agent would have to satisfy these twelve corollaries to be considered conscious. The canonical architecture of the WiSARD model does not allow the construction of a system that meets these requirements, so this thesis will investigate how to extend this model to advance research on the construction of a weightless cognitive agent. The creation of such an agent is not within the scope of this research, this thesis is limited to the creation of some of its components only.

Panella et al.[21] points out as the current limitations and open research questions of cognitive architectures the following items: (i) integration of distinct cognitive functions; (ii) computational efficiency of the platform; (iii) parallel processing of data; (iv) integration of heterogeneous knowledge representation; (v) integration of heterogeneous reasoning and cognitive functions; (vi) integration of planning, acting, monitoring and goal reasoning; (vii) knowledge problem (knowledge acquisition, knowledge size, and homogeneity of knowledge). One of the main points is related to the computational cost of such cognitive architectures so that an agent with a weightless cognitive architecture can offer a significant contribution in this regard, given the low computational cost of the WiSARD model.

1.2 Positioning WiSARD in Relation to Other ML Models

This section will situate the WiSARD model to mainstream machine learning models to highlight its idiosyncrasies and justify its choice as the basis of a cognitive system. The main features of WiSARD are:

- its speed in training time, without the need for any type of technique that involves searching for convergence, so that this model can perform online learning;
- the ease of implementing forgetting mechanisms;
- the possibility of low-cost implementation in terms of memory when using sparse data processing structures;
- a huge VC-dimension[31];
- its simplicity of hardware implementation;
- the ease of viewing patterns learned in each class;
- its ability to learn with few examples, presenting a rapid growth of its learning curve in many domains;
- its facility for parallel training.

WiSARD originally is only capable of classifying binary inputs, so this model is highly dependent on the preprocessing of inputs. Although classification is an important task in machine learning, many tasks require a model with the ability to handle a much higher level of complexity. The online learning ability of weightless neural models, due to their short learning/classification times and low computational cost, are major advantages of this type of ML model, make it an interesting complement to today's deep learning-dominated ecosystem, which despite its tremendous contribution to the entire ML area, due to its ability to handle invariance, has remarkable limits and little effort has been made to propose new alternative models with light requirements[32].

In the last decade we have seen significant progress in several areas of machine learning, such as image classification with noisy backgrounds[33], reinforcement learning and its use in many complex games[34], the use of transfer learning to accelerate training and transfer knowledge between domains[35], artificial creation of images[36], and audio[37], video[38] and text[39] through generative adversarial networks. Other advances include CapsNet and its capacity to deal with variation in diverse features[40], advances in LSTM and its use for speech recognition[41] and simultaneous translation from speech to text[42] and the creation of YOLO and the advance of object detection[43].

Despite the great advances in ML caused by the rise of deep models in the last decade and their undeniable historical contribution, these models also have visible limitations that are often overlooked in favor of hype. Some fundamental limitations of deep models are listed below.

- **Difficulty in learning logical rules:** many researchers have pointed out the difficulty of reconciling deep models with a neurosymbolic approach that takes advantage of the accumulated knowledge of a given domain[44, 45]. In fact, some experiments point out how humans have an advantage in this type of task compared to DNNs[46];

- **Transfer's capacity:** the canonical architecture of DNNs makes it difficult to transfer knowledge to other domains[47–49]. Some experiments show that features extracted by DNNs have less representative capacity than might appear at first glance[50];

- **Limited understanding of architectures:** the architecture of DNNs limits its ability to generalize when the hierarchical structure of the data in the training set is far removed from the test set, as [51] shows to RNNs and complex linguistic constructions;

- **Inability to deal with open-ended inference:** DNNs isn't able to extract implicit knowledge from the data, despite [52, 53] make an initial move in this direction;

- **Lack of transparency:** due to its large number of parameters, the majority without a topological semantics, DNNs have a black-box structure, being difficult to interpret[54–57];

- **Necessity for a large dataset:** although data augmentation techniques have allowed DNNs to perform well on small datasets, DNNs have steep learning curves and this is perhaps the limitation most evident by the community consensus. Zohuri & Rahmani[58] explains in detail this characteristic of DNNs, which is enhanced by their lack of common sense.

These were limitations of DNNs classical models and all of them were significant restrictions on deep models when this research work began to be developed.

Many advances have been made since then, minimizing these obstacles, such as the improvements in explainable model-agnostic algorithms that has considerably increased the interpretability of DNNs[59] or the increased usability of deep models through incremental learning[60, 61].

Due to its peculiarities that will be presented in details in Chapter 2 and explored in other chapters, WiSARD is both (i) a viable alternative to DNNs in tasks that demand online learning, light memory requirements, and hardware implementation (economy of resources; high parallelism; no need to update weights in training, thus saving all the operations in which this implies), and (ii) a suitable partner in the composition of hybrid systems, creating architectures that combine the advantages of WiSARD and DNNs, as has already been explored[62, 63].

Despite the prominence of DNNs, other mainstream ML algorithms have continued to be researched in the past decade, but they all have well-known limitations: (i) although CART has a high potential of interpretability, is competitive with state-of-the-art in many tasks, and is non-parametric, these models usually become overly complex with few layers[64]; (ii) SVM has as advantages its relative efficiency in memory and its effectiveness in higher dimensional spaces, however, it is not suitable for large datasets, it is not robust against noise and it does not perform well when the number of data features exceeds the size of the training set[65]; (iii) Naïve Bayes is able to perform well with few samples in the training set, however, its effectiveness is limited to the degree of independence of the features and realistic datasets tend to have some degree of correlation between their features. This problem can be smoothing with proper data analysis, but it still imposes a strict limitation on the model[66]; and (iv) kNN does not have a training stage, thus saving training time and allowing new training data to be added at any time, without the need for retraining. On the other hand, this peculiarity of kNN makes it dependent on memory to store the entire training set. kNN is also vulnerable to noisy, missing values and outliers, in addition to not performing well in high dimensions[67].

WiSARD is uniquely positioned in the face of these algorithms: (i) WiSARD is non-parametric like CART and kNN, while preserving its simplicity and is inexpensive in terms of memory consumption; (ii) WiSARD is memory efficient like SVM, while its latest versions are robust to overfitting large datasets; (iii) WiSARD has online learning as well as Naïve Bayes, being able to deal with interdependence of the features if there is an adequate representation for the data; and (iv) WiSARD is vulnerable to noise as well as SVM, Naïve Bayes and kNN, however, due to the interdependence of training different patterns on WiSARD it is possible to implement internal mechanisms in it that increases its noise tolerance, complementing the data analysis smoothing process. This comparison is better visualized in Table 1.1.

In addition to its simplicity, efficiency, and good accuracy in many tasks, an idiosyncrasy that makes WiSARD an interesting model today is its preservation regardless of the trained patterns that allow greater control over the learning process based on topological adjustments and adequate preprocessing.

One of the first steps necessary to expand the scope of action of WiSARD can be achieved by enabling the model with the ability to provide continuous output, that is, adapting it for regression tasks. In addition, many recent domains require the model to be able to detect the simultaneous occurrence of multiple classes, which is known as multi-label classification. In many tasks, it is also necessary to combine different specialists, in order to obtain a system that is more robust than individual classifiers, which is known as ensemble learning. Since all of these WiSARD extensions are possible directly from minor modifications to the original model, they were the goal of this work.

1.3 Some Thoughts on Historical Development of ML Models

In his famous analysis of paradigmatic changes in "The Structure of Scientific Revolutions"[89], the philosopher of science Thomas S. Kuhn points out that when a number of anomalies in a given scientific area become large enough to be ignored, a revolution in the current explanatory model is needed. And when a new model capable of satisfying such anomalies is chosen as appropriate by the community consensus, there is a shift in scientific work towards this new paradigm. The work of normal science will then be to perfect this paradigm, increase its area of application and close the gap between the empirical evidence and the theory that underlies the new paradigm. Throughout this great undertaking, new anomalies will arise and will accumulate until they lead to the point of a new scientific revolution. When this happens, an outdated model, ineffective for satisfying the new evidence collected in a more appropriate way, is set aside by the community.

It would be a mistake to consider that the shift from the machine learning community to the deep learning paradigm is the edge of such a revolution since deep models have not solved all the anomalies and open questions that have accumulated in the community historically, nor have they defeated competing models completely. In fact, despite having provided high-performance solutions in many tasks and establishing new state-of-the-art in many domains, deep models immediately created new issues, in addition to not addressing all the old ones, as we already mentioned in Section 1.2.

It would be wise to conclude, therefore, that deep learning has developed along the path of normal science, as more of a community effort to find answers to funda-

Table 1.1: A comparison between WiSARD and other ML models

Model	**Speed**	**Memory**	**VC dimension**	**Hardware**	**Interpretability**
WiS ARD	Fast training; online learning; time depends on binarization	Low memory consumption (using sparse structures representation)	$d_{VC} = N(2^n - 1) + 1$ [31]	Efficient, using hash[68, 69]; FPGA implementations[70]	High
DNN	Training tends to be slow when transfer learning is not used	High; is required to store input, weights and parameters[71]	From $O(E)$ to $O(E^2)$, being E the number of edges; this is small, despite its learning potential; LTH explains this[72]	High computational cost; acceleration techniques turn it viable[73]	Low, due to lack of topological semantics; except for xDeep[74]
CART	Fast, dependant on the number of features	Uses too large heap space for constructing trees; can be mitigated using strict policies[75]	$2l \geq \binom{d}{\lfloor \frac{d}{2} \rfloor}$; being d the length of the greatest tree and l the number of leafs. [76]	Simple and efficient[77]	High[78]
SVM	Slow due to kernel trick; can be miti gated by using less training data and Linear SVM[79]	Memory-intensive in training, when working in the dual space[80]	$n + 1$ being$n$ the number of hyperplanes[81]	Need special dedicated architectures due to its computational cost[82]	Through fuzzy rules[83]
Naïve Bayes	Fast; online learning	Low	$\|\chi\|$; being χ the number of parameters[84]	FPGA using stochastic discrimination theory [85]	High; due to the features indepen dence assumption [86]
kNN	No training stage; saves training time and allows adding new training data without retraining	kNN depends on memory to store the entire training set	$VCdim(H) \geq k$ [87]	Acceleration is necessary for applications with massive high dimensional data[88]	High

mental credit assignment problems[90] than a universal solution for them. It is not even a solution so strong that it excludes the contribution of other models in the process of paradigmatic evolution.

In fact, as the DNN pioneer Jürgen Schmidhuber exposes in his historical recapitulation of DNNs[91], the components that would have constituted the hyped phenomenon of deep models in the first decade of the 2000s had been developing for decades before: (i) the minimization of errors through gradient descent[92] in the parameter space comes from the 60s[93–101]; (ii) explicit backpropagation mechanisms in arbitrary, discrete, possibly sparsely connected networks in the early 1970s[102, 103]; and (iii) convolutional filters debuted on Neocognitron[104, 105] in the late 1970s.

Likewise, although n-tuple classifiers are based on an ancient method, more than six decades old and which has remained outside the mainstream efforts of the community in recent years, this should not be enough to ignore its numerous contributions and potential innovations. Even more than developments in the area of hardware, as well as progress in the mathematical understanding of the model, can powerfully enhance its qualities soon, as will be discussed in Chapter 2 of this thesis.

In a recent survey[106] on n-tuple classifier subspace, the authors point out: "The n-tuple method and its extensions are appropriate for problems even with 1000 dimensions and 100 classes for which it or its extensions may be the most economical online classification method that trades off computational complexity with fast memory, the cost of which has become cheaper and cheaper over the last decades."

1.4 Contributions

This thesis explores the development of extensions of the WiSARD model and new ways of grouping these neural networks into ensembles. The main contributions of this work are listed below. This thesis uses a map-based WiSARD implementation, instantiating only the memory locations that were actually trained (the complete description is given by Appendix G.

- A reinterpretation of the multi-label approaches Label Powerset and Binary Relevance taking advantage of WiSARD's multi-discriminator architecture;
- New weightless ensembles based on voting systems, whose use of ballots is directly related to the WiSARD architecture;
- An enhanced version of the n-tuple regression network using different kernel

estimators, in order to better approximate the probabilistic density function according to the dataset;

- A new weightless regression model that uses multiple n-tuple regression networks, in order to approximate pdf according to the local behavior of the regression function.

1.5 Thesis Outline

Chapter 2 presents the n-tuple classifier, the first machine learning model based in n-tuple pattern matching. It also presents the WiSARD model, its extensions, and its applications in various machine learning tasks. In this chapter, several models of weightless neural networks are also introduced. The chapter concludes with a discussion of the main limitations of these models, as well as what are their main advantages and uses in modern machine learning scenarios. This chapter presents too the preprocessing techniques used in this work. Since WiSARD deals only with binary inputs and the preprocessing of data directly imply the performance of this model, the choice of binarization is a vital element in the WiSARD pipeline and, therefore, an entire chapter has been reserved to detail the techniques used here. Preprocessing of categorical and quantitative variables, image, audio, and text are presented in this chapter.

From Chapter 3 we have the contributions of this thesis. Since my master's thesis[107] was based on the application of WiSARD in classification of facial emotions, this work started with the exploration of WiSARD in classifying Action Units, the facial muscles used to express emotions. Chapter 3 presents two new WiSARD-based multi-label architectures and discusses their validation in the task of classifying Action Units. The systems described in this thesis are related to two important characteristics of architectures based on the Fundamental Neuroconsciousness Postulate: emotion and perception.

Since each approach introduced in Chapter 3 had advantages and disadvantages, the need to explore ways to combine the idiosyncrasies of WiSARD-based solutions became explicit. Chapter 4 introduces five new models of WiSARD ensembles proposed for this purpose. This chapter presents their validation in several datasets with different domains: numerical data, images, and text.

In parallel to the development of the research presented in Chapter 4, attempts have been made to extend WiSARD for regression, since many ML domains require models that deal with continuous data. Chapter 5 presents the two WiSARD-based models proposed for regression, its ensembles, and its exploration in diverse datasets. This chapter also details the WiSARD-based systems used in a palm oil prediction challenge, with one of these solutions reaching one of the highest positions in the ranking, with a difference of four orders of magnitude in the training time of the winning model. Since prediction is an essential attribute of a system that aims to satisfy the Fundamental Neuroconsciousness Postulate, the contribution of this chapter fulfills one of the most important requirements for obtaining a weightless cognitive agent.

In order to apply all novel WiSARD-based models to a real-world problem, Chapter 6 deals with the affective computing task presented in Chapter 3, the ensemble architectures presented in Chapter 4, and the regression models developed in Chapter 5 in a task of multi-modal prediction of empathy. This chapter offers an analysis of WiSARD extensions and their ensembles performance on empathy prediction and also explains the composition of the binary input obtained by combining image and audio features.

Chapter 7 summarizes the thesis and discusses the limitations of the proposed models and techniques. This chapter also discusses possible explorations and applications of the WiSARD model in general. Finally, this chapter discusses the WiSEMAN cognitive architecture as a framework for multi-agent systems by using all the extensions of the WiSARD model developed in this thesis.

Appendix A shows the complete results related to Chapter 3. Appendices B and C show the complete results of the experiments described in Chapter 4 and discusses the relationship between the number of weak learners and the number of training partitions with the accuracy of each ensemble. Appendix D shows additional results to those present in Chapter 5.

Appendices E and F debates the Fundamental Postulate of Neuroscience, its corollaries, and how the regression-based model presented in Chapter 5 can contribute to a discussion on the topic of artificial consciousness, by bringing the WiSARD model even closer to the fulfillment of the requirements derived from this postulate. Additionally, an architecture based on WiSARD and concepts presented throughout the thesis for organizing multi-agent systems is proposed.

Appendix G presents the code libraries used in this thesis. Appendix H lists all papers accepted for publication from the content of this thesis.

Chapter 2

Weightless Artificial Neural Networks

In this chapter, we will discuss the background, the main features, recent work, and some promising possibilities in relation to the WiSARD classifier.

2.1 n-Tuple Classifier

The n-tuple classifier is a binary pattern classifier[1] based on memory array, requiring no parameter fine-tuning or any error minimization technique to achieve generalized learning patterns. The basis of its operation is to use the input to construct an address set and use them to access the memory contents. In this way, the n-tuple classifier is formed by a matrix, where each column is associated with a class and the lines correspond to combinations of bits of the input. This model is illustrated by Figure 2.1.

Thus, in this model, the training phase consists of writing in memory locations, while the classification phase in reading the same locations. This model was proposed by Bledsoe & Browning in 1959. In 1962, Bledsoe & Bisson[108] proposed ways to make the n-tuple pattern recognition method more efficient, through changes in training policies and using tiebreaker techniques in the classification phase. Models

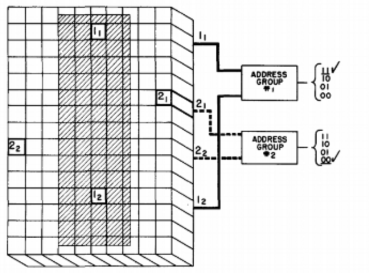


Figure 2.1: The n-tuple classifier (extracted from [1]).

based on n-tuple classifier are commonly called Weightless Artificial Neural Networks (WANNs).

2.2 WiSARD

WiSARD[6] is a n-tuple classifier composed by class *discriminators* proposed by Igor Aleksander in 1969. In this model, each discriminator is a set of N Random Access Memories (RAMs) nodes having n address lines each. All discriminators share a structure called *input retina*, from which a pseudo-random mapping of its $N * 2^n$ bits composes the input address lines of all of its RAM nodes. Due to the fact that it separates the learning of different classes in different artifacts, WiSARD is more modularizable than the original n-tuple classifier.

The biological analogy behind this model lies in the mapping of the synaptic strength between the output produced and transmitted by the neuron's axon and the input of a post-synaptic neuron, into pseudo-continuous numerical weights. An important simplification happens in the way inputs to neurons are modeled: all synaptic connections terminate directly at the neuron's soma. Although such specific morphological arrangement is plausible in biological terms, the vast majority of synapses in the central nervous system terminate at the neuron's dendritic tree. Nevertheless, generalizations of artificial weighted-sum-and-threshold neurons, such as Sigma-Pi units, do exist, this means that the dendritic tree, the most noticeable morphological structure of the neuron cell, is not being taken into account in mainstream artificial neural network paradigms[7].

2.2.1 Training and Classification

When the network is initialized, all RAM memory locations have "0" as content. In its original definition, when receiving a binary training input, WiSARD set to "1" the contents of the memory locations accessed in the discriminator of the sample class (Fig. 2.2). In the classification phase, all discriminators have each of their RAMs accessed in a single memory location, and the discriminator who has accessed more memory locations with content "1" will determine the class of the example (Fig. 2.3).

The original model suffered from saturation as the cardinality of the training set increased. To circumvent this limitation, in an enhanced version WiSARD[109] happened to have a counter in the positions of memory of the RAMs, increasing its value to each access during the training. During the classification phase, it continues to count the number of memory locations with non-null content to determine the score of each discriminator, however, a memory location can only be counted if its

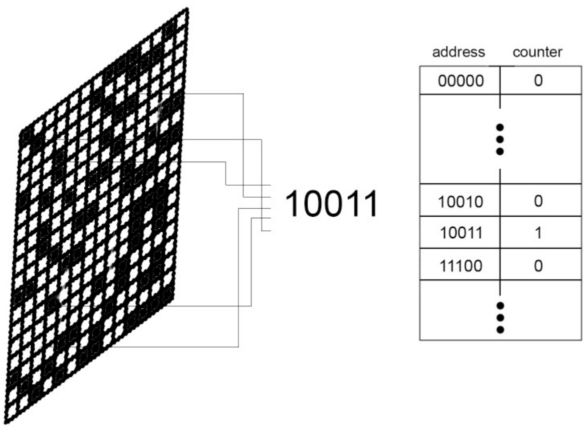


Figure 2.2: Training stage in WiSARD[2]

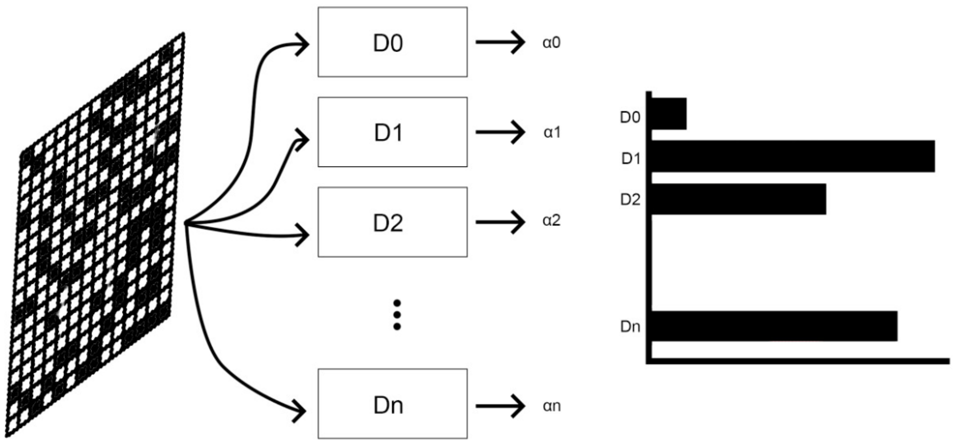


Figure 2.3: Classification stage in WiSARD[2]

content has a value greater than a threshold called *bleaching*, which is initialized with value 0. When there is a tie between the discriminators, the *bleaching* is increased. If the value of the bleaching becomes greater than the counter of the most accessed memory location of WiSARD, there is an absolute draw and a default class chosen beforehand is determined for the sample. A more elaborate mathematical formulation of the training and classification of the n-tuple classifier and WiSARD can be found in literature[31, 106].

A facility promoted by training oriented to writing in memory is that learning of each pattern preserves a type of independence, and it is very simple for the model to forget a certain pattern: when receiving a pair <binary input, label>, just use the label to access the corresponding discriminator and use the binary input to access specific memory locations from this discriminator, subtracting by one the counter stored in this memory location. Such algorithm considerably reduces the implementation of cross validation and, especially, leave-one-out validations.

In WiSARD, the base of the tuple does not necessarily need to be 2, since in some domains certain possibilities for variations of the features can be related to the values of the bits, adding the tuples semantically. While this method has the advantage of using domain information directly in the formation of tuples, on the other hand, it causes more memory locations in the RAMs, further increasing the sparse network. An example of a task in which a WiSARD with a base other than 2 was used is in a competition to create intelligent rock-paper-scissors players, where the network was used with base 3[110].

2.2.2 Mapping

Although the WiSARD mapping is normally pseudo-random, it can be changed depending on the domain, if it is desired that certain bits of the input form the same tuple, if there is a semantic link between them. The mapping can also be replicated to generate oversampling and if the number of bits in the input does not allow an exact division in the number of tuples, the mapping can disregard a bit or repeat as many as necessary in new tuples[108].

Optimization techniques for WiSARD mapping that have been developed throughout its history include: genetic algorithm with simple mutations[111], crossover between the mappings[112], tuples with different sizes and probabilistic mechanism for selecting the size of the tuples (in images the probability is calculated according to the distance of a given pixel to the center of the image)[113], stochastic search using a reward and punishment system for the performance of individual tuples[114] and particle swarm optimization of n-tuples[115].

Guarisa[116] introduces other approaches to mapping: (i) particle swarm optimization of mappings; (ii) an improved version of the genetic algorithm proposed by Giordano & Massimo[112], adding diversity to the initial population and using values of the objective function in the selection of individuals; (iii) creation of a mapping based on the ordering of the input bits in relation to the entropy level of the access probabilities of these same bits in mental images (see next subsection, Section 2.2.3); (iv) the use of weighted constraint satisfaction problem[117] to separate bits '1's in separate tuples in sparse datasets, in order to balance the RAMs. This latter approach is the only one that is based directly on the examples of a particular dataset.

2.2.3 DRASiW and Mental Images

An interesting feature of WiSARD discriminators is that as the mapping of the input in memory locations is known, it is possible to apply a reverse engineering method to the memory contents of the RAMs to generate a new input, which represents the pattern learned by a specific discriminator. This process is known as DRASiW[109] and the generated inputs are known as mental images.

Mental images are gray-scale inputs with the same dimensions as the WiSARD retina, and access counters stored in memory locations are used for their formation. The input bits that generated the most accessed locations will be filled with RGB(255, 255, 255), while bits that generated never accessed locations will be filled with RGB(0, 0, 0). All other locations are filled with shades of gray from a normalization of the counters.

In image datasets, this is a good way to observe if there is any type of noise affecting learning if there is saturation in a specific discriminator or even empirically visualize the main features that allow the correct classification of each class.

2.2.4 minZero and minOne

Another modification in the WiSARD algorithm is to ignore the contribution of a certain memory location in the classification phase if its address does not meet a certain amount of "0"s (minZero) and "1"s (minOne). The motivation for these parameters is to be another way by which the WiSARD can filter redundant input data, besides the necessary preprocessing applied to the input (via binarization). There is still no way to dynamically adjust minZero and minOne, and the best values for them vary for each task and are empirically obtained through a simple exploration of the value space.

2.2.5 VC-dimension

In ML theory, the VC dimension[118] is a measure of the capacity of functions that can be learned by a binary classifier, that is, the cardinality of a set of data that can be learned without saturation. The exact WiSARD VC dimension was calculated for the following cases:

- traditional WiSARD with a single discriminator: $d_{VC} = N(2^n - 1)$, where N and n are the amount of nodes and the addressing tuple length, respectively[119]. In this case, this WiSARD has only one discriminator that will classify a sample by comparing its score with an acceptance threshold;
- traditional WiSARD with two discriminators: $d_{VC} = N(2^n - 1) + 1$[31];

- WiSARD with discriminators storaging counters and using *bleaching*: $d_{VC} = N(2^n - 1) + 1$[31].

2.2.6 Scalability

Some possible methods to train the WiSARD method in a scalable way include:

- in the training of each sample, train each RAM independently and simultaneously;
- separate the training dataset by class and train each discriminator independently and simultaneously;
- create several models with the same mapping and addressing and divide the training set into all of them and then unify them into a single WiSARD, whose content of each memory location is the sum of the counters of the networks used for training, performing a kind of federated learning.

2.3 ClusWiSARD

A version of WiSARD was created to deal with unsupervised data, based on the Fuzzy ART1 networks[120], which represents each class as a node, which is a vector of binary inputs. Fuzzy ART1 has the ability to dynamically allocate new nodes, if necessary. This modelis called AutoWiSARD[121, 122] and aims to solve the stability-plasticity dilemma[123], which tries to find the optimum amount of elements in a cluster, so that it is not too heterogeneous, nor excessively restrictive. Basically, AutoWiSARD works as a WiSARD that is started with a single discriminator, which has two thresholds. When it receives an example for training, she will rate it and use the score obtained to decide whether to train the example or not. If the example score is between the window formed for both thresholds, the discriminator learns the example, since it is located in its stable zone. If its score is below the lower threshold, a new discriminator will be created to learn it, since this example is located in its plastic zone. If its score is higher than the upper threshold, this example will not be learned, as it is considered sufficiently similar to the previous ones, whose standard is already defined in the discriminator.

Sometimes the same class must include non-similar patterns. Not differently from other classifier models, WiSARD's discriminating capabilities will be stressed, probably inducing the target discriminator to become saturated due to the learning of extremely heterogeneous patterns. ClusWiSARD [124] is an extension of WiSARD, which allows it to learn sub-patterns by allowing the creation of more than one discriminator per class if the new examples submitted to the network for learning are not sufficiently similar to those already learned. Since this is analogous to clustering the examples of a class, the network was called ClusWiSARD and this operation is called internal clustering. Unlike AutoWiSARD, ClusWiSARD can also be used for supervised learning, as well as unsupervised.

In the training phase, when an observation is sent to the network, it is sorted by all the discriminators in its class, which will naturally return a score with the number of active RAMs during classification. The discriminator that will learn a new pattern must satisfy the following condition:

$$r \geq \min\left(N, r_0 + \frac{N|d|}{\gamma}\right), \tag{2.1}$$

where r is the score of the discriminator when classifying the observation, $|d|$ is the discriminator size, N is its number of RAMs, r_0 is a threshold, which indicates the minimum response expected by a discriminator, and γ is also a threshold, which indicates the growth interval, that is, the speed that the discriminators increase their size.

The classification phase of ClusWiSARD is similar to that of WiSARD, where the discriminator with the highest score will determine the pattern of the example. If there is a tie between discriminators of the same class, this class will naturally be the network response and there is no need to apply *bleaching*.

Although originally designed to improve accuracy in supervised tasks, by enabling learning of sub-patterns, ClusWiSARD can also be used for semi-supervised learning (where a non-annotated example is trained by all the discriminators that present the highest score in the classification phase, being able to belong to more than one cluster per class and to more than one class) and unsupervised learning[125] (where a ClusWiSARD presents only one discriminator and applies the same policy of creating new discriminators of supervised learning, one example being always tested for all discriminators already created; this operation is known as external clustering).

In a challenge of financial credit analysis, ClusWiSARD outperformed SVM by two orders of magnitude in training time, while remaining competitive in accuracy[124]. Data stream clustering is a challenging task, as it involves dealing with data that has become obsolete, and in many tasks in this domain there is data coming from multiple sources, which almost always implies different features. WiSARD has proven successful in this field, mainly through its ClusWiSARD extension[124, 126], that is detailed in Section 2.3, adapted to include forgetfulness mechanisms. Results obtained can be found at [124, 126–129].

2.4 KernelCanvas

Time-series classification has a wide range of practical applications like speech recognition, but one big challenge in tackling this kind of problem is data length, which is not always the same. This turns out to be a problem for many models that require data to have equal dimensions. KernelCanvas is a method that enables WiSARD to deal with spatio-temporal data by transforming it into a fixed-length binary input.

The KernelCanvas method is based on the action of using a paintbrush on a white canvas. When drawing a figure like a character on the canvas, it does not matter how many times the brush stroke the canvas or the order the figure was drawn, since the canvas itself will preserve its dimensions. In a similar way, KernelCanvas makes use of a set of kernels to represent partitions of the canvas.

A time-series dataset sample can be seen as an array of variables measured during a certain period of time. The elements of this array can contain any dimension but one might consider an array of two-dimensional data for the sake of explanation. It takes the (x, y) coordinates of the pixels of a handwritten character. Each point indicating the part of the image that was drawn at some instant.

The empty canvas can be interpreted as a zero matrix and the kernels are points in the canvas that address groups of indices of that matrix. The kernels are usually generated in a uniform space $[-1, 1]$, yet there is no restriction on this bound. It implies that the data needs to fit this space and some kind of normalization must be applied. For a given sample, all points in the array must be compared to all kernels using some metric, such as the euclidean distance. The kernel that presents the shortest distance to the point is then activated so that all bits addressed by this kernel are set to 1. After all points in the sample are computed, the canvas represents a binary pattern that can be applied to the weightless model.

Some applications of KernelCanvas include:

- **Time-series classification:** Spatio-temporal data add new dimensions to typical classification problems since the way in which the input is formed is itself one of the most important features of the inputs. For WiSARD this is a special problem that is difficult to solve since context windows can result in data of different sizes, and in WiSARD the size of the entry is fixed due to the retina. In addition, resizing the input adds noise and depending on the number of times this is required in a task, the classification capability of the WiSARD rating can be completely compromised. Combined with KernelCanvas, WiSARD and its extensions proved robust when dealing with handwritten character recognition, speech recognition[130] and empathy prediction in videos[131].

- **Audio Processing:** Music tracking is a technique that allows you to find out which part of a song is being played at any given time. Representing this task as a classification problem implies treating each piece of the original audio as a class, thus generating many classes and few examples in each of them, making this an interesting case for the use of WiSARD, given its ability to generalize with few samples. WiSARD, combined with KernelCanvas and Markov Localization Algorithm[132], proved to be efficient for this task, as detailed in Souza et al.[133].

2.5 n-tuple Regression Network

n-tuple Regression Network[134] is a modification of the basic n-tuple classifier architecture, which allows it to operate as a non-parametric kernel regression estimator; it is also capable of approximating probability density functions (pdfs) and deterministic arbitrary function mappings; for this, the n-tuple Regression Network uses a RAM-based structure, where each memory location stores a counter and a weight, which is updated through the least means square algorithm [135].

Non-parametric regression estimates a function directly, unlike parametric regression, which estimates the parameters of the approximating function. For this, two assumptions are made: that the function is smooth and continuous[136]. In cases where the relationship between the dependent variable and the explanatory variables is unknown, non-parametric regression is more appropriate, as it can be adjusted to capture unusual or unexpected features of the data.

This being a scenario found in many tasks where ML is regularly applied, it is natural that a neural network for regression is oriented to work with non-parametric regression. This is the case with the extension of the n-tuple method for regression, whose construction and mathematical foundation will be analyzed below.

2.5.1 General Regression Neural Network

For regression tasks, it is assumed that any input $\mathbf{x}$ and output y of the system is associated with the random variables $\mathbf{X}$ and Y, respectively. The system will estimate a conditional mean of the dependent variable Y for any value of $\mathbf{x}$, considering that:

$$m(\mathbf{x}) = E(Y|\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x}) \tag{2.2}$$

It is assumed that the conditional mean exists and is well defined on the domain of the input. For a known underlying probabilistic density function (*pdf*), the regression function is given by:

$$m(\mathbf{x}) = E(Y/\mathbf{x}) = \frac{\int_{-\infty}^{\infty} y \cdot f(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} f(\mathbf{x}, y)dy} = \int_{-\infty}^{\infty} y \cdot f(y|\mathbf{x})dy \tag{2.3}$$

However, when there is no explicit knowledge about the system, it is necessary to estimate the regression function from a finite set of known points. Non-parametric regression methods perform this estimation, without any need to make assumptions about the shape of the regression function. Kernel methods are among such non-parametric methods and are based on applying a smooth monotonically decreasing, non-negative and continuous function for each pair $< \mathbf{x}, y >$ taken from the distribution to estimate the *pdf*[137–141].

It is convenient to choose a $(D + 1)$-variate kernel function, in such a way that this is separable to $\mathbf{x}$ and y:

$$\phi(\mathbf{x}, y) = \phi_x(\mathbf{x}) \cdot \varphi_y(y), \tag{2.4}$$

where $\phi_x(\mathbf{x})$ and $\varphi_y(y)$ are univariate kernel functions.

The regression function will be given by:

$$\hat{E}(Y|\mathbf{x}) = \frac{\sum_{i=1}^{T} \phi_x(\mathbf{x} - \mathbf{x^i}) \cdot \int_{-\infty}^{\infty} y \cdot \varphi_y(y - y^i)dy}{\sum_{i=1}^{T} \phi_x(\mathbf{x} - \mathbf{x^i})} = \frac{\sum_{i=1}^{T} y^i \phi_x(\mathbf{x} - \mathbf{x^i})}{\sum_{i=1}^{T} \phi_x(\mathbf{x} - \mathbf{x^i})} \tag{2.5}$$

The estimator in Equation 2.5 is known as Nadaraya-Watson[142, 143] and it can be interpreted as the weighted average of $Y_1, ..., Y_n$ given a set of weights $W_i(\mathbf{x})_{i=1}^{n}$.

A General Regression Neural Network (GRNN)[144] is a model capable of performing non-parametric regression from a training set only, which can be relatively small. GRNN stores the entire dataset, based on this data to estimate the system's *pdf*.

There are many types of GRNNs and, despite their differences, they work the same way: they store the entire training set and select a kernel function and a smoothing parameter from it. The major limitation of this model is the memory cost to store the entire training set and Kolcz[145] uses an n-tuple neural network to approach a GRNN, taking advantage of the fact that its access counters can be analogous to an implicit type of storage of the training set.

2.5.2 The Approximation-Type n-tuple Neural Network

To approximate a GRNN with an n-tuple neural network, Kolcz & Allinson[134] used a variation of the model where each memory location stores two different contents: access counters and weights. These weights are updated using the Least Means Square gradient descent algorithm, as is done in other single layer architectures since this minimizes the weighted least-square error. Such weights are initialized with the value 0 and at each access during the training phase, they are increased by the value of the dependent variable associated with the trained input.

Since the memory locations accessed by an input during the prediction phase have their counters related to the tuples formed by the $\mathbf{x}$s of the training sample and their weights formed by their ys, Equation 2.5 can be applied here to generate the prediction of the n-tuple neural network, that is, a Nadaraya-Watson estimator can be approximated by simple arithmetic mean of the sum of the weights and the sum of the counters accessed.

However, for this to be true, the set of hit counters accessed must be the distance between the $\mathbf{x}$ that is being predicted and the $\mathbf{x}$s of the training set, that is, that the sum of the counters of the accessed memory locations is equal to $\sum_{i=1}^{T} \phi_x(\mathbf{x} - \mathbf{x^i})$. [134] define the distance between two network inputs as the number of different tuple addresses they generate. Many analyzes[146–148] have suggested that the tuple distance is equivalent to a generalized Hamming distance.

This is used to calculate how many memory locations accessed in a prediction were never accessed during training, that is, how many different tuples exist when comparing two inputs. Therefore, the kernel in an n-tuple neural network will be obtained easily by the sum of the counters stored in memory locations addressed by tuples that are identical (Hamming distance equal to zero) between each example of the training set and the input whose associated y is being predicted. In this way, there is no need to store the entire training set as in a GRNN. Finally, in this model the smoothing parameter is the tuple size.

2.6 Other Weightless Artificial Neural Networks

Other WANN models include:

WiSARD's alternative implementations:

- **WiSARD PUF:** Physically Unclonable Functions (PUFs) are circuits that use one-way functions which map inputs to unique outputs[149]. Ideal PUF chips have three characteristics: security, uniqueness, and reliability. Recently, many practical applications based on PUF have been shown to be highly vulnerable to attacks made by machine learning algorithms. In order to increase the security of such chips and improve its uniqueness, new PUF models were created based on the WiSARD architecture.

  WiSARD PUF[68, 69] is a collection of RAMs, not necessarily of the same size. In this model, the memory locations are the same as in the original WiSARD, storing only bits and not counters. As in the other n-tuple classifiers, WiSARD PUFF uses a pseudo-random mapping between the input bits and the addresses of the RAMs, but here the tuples can be of different sizes depending on the RAM. The training is similar to traditional WiSARD. And when the network receives a challenge string it enters a mode equivalent to the traditional classification, only here there are two different counters for the "0"s and "1"s accessed. The response from WiSARD PUF will be equal to the bit most accessed at this stage.

- **AMQ Filters-based WiSARDs:** Since the original implementation of WiSARD, as well as those based on dictionary-like structures, requires a considerable amount of memory resources to achieve good learning features, new implementations[150, 151] have been proposed based on Approximate Membership Query (AMQ) structures[152], probabilistic data structures that use less space than dictionaries. Four types of AMQ-structures were tested: Bloom Filter[153, 154], Cuckoo Hashing[155], Cuckoo Filter[156] and Quotient Filter[157].

A WiSARD-based multi-layer model called **NC-WiSARD**[158]. One of the weaknesses of the WiSARD lies in its inability to deal with invariancy, since when the inputs of the training set are binarized if there is variation in relation to rotation or scale in specimens of the same class this will result in different patterns overlapping and causing great noise in the content of discriminators. To circumvent this limitation NC-WiSARD was created. This weightless neural network is based on AutoWiSARD and the Neocognitron network[159], a hierarchical feedforward neural model inspired by the visual cortex.

NC-WiSARD has three types of cells per layer:

- D cells: responsible for connecting to the previous layers;
- S cells: responsible for selecting a subset of the characteristics of a feature of the previous layer;
- C cells: responsible for applying distortions to the selected features by S cells.

PLN[160]: While weightless models traditionally have one layer, Probabilistic Logic Node (PLN) is the first multilayer weightless approach. In this model, each memory location stores a single bit, which can be "0", "1", or "u". The network is shaped like a pyramid, that is, the last layer has only one RAM. When RAM is accessed, it propagates the bit stored in the memory location in which it was accessed to the next layer. The entire network is initialized with the "u" bit. In the training phase, when a pattern is submitted to PLN, each bit "u" is accessed and makes a bit "0" or "1", with the same probability. If there is convergence, this pattern is learned, otherwise, the process is repeated.

This model received two extensions: m-state PLN (MPLN)[161], where each memory location can store discrete values and the output can operate as a linear or sigmoid function, and pRAM[162], where each memory location can store continuous values in a range [0, 1].

GSN[163]: One of the great problems of PLN was its high rate of saturation and corruption of previously learned data since when a new example was submitted for learning, there were as many iterations in the pyramid as were necessary for there to be convergence and, finally, learning the new pattern.

To overcome this limitation of the previous model, a new one was proposed: Goal Seeking Neuron (GSN). The novelties of this model are listed below.

- In GSN RAMs propagates "u" as a signal to the next layer and even is possible for the network to have an "u" output.
- When a RAM is accessed in a memory location with "u" content, it propagates two distinct signals ("0" and "1"), accessing two memory locations in the RAM that are addressing in the next layer.
- In the validation phase, if there is no success, learning a certain pattern is abandoned so as not to cause saturation in the model.
- If the network output is "u" during validation, this indicates that the network is ready to learn any desired pattern.
- If the pattern is learned and the RAM of the last layer has been accessed in two memory locations that contain a bit that allows the pattern to be learned, then one of the memory locations is randomly drawn to be used to perform the recall where the writing will take place the new contents of the previous layers.
- If there is access to memory locations storing "0" and "1" in the last layer, the network output is "u".

GRAM: Since in PLN and its derived models, neurons do not have the ability to generalize, something that is only possessed by the network, a new model was proposed to circumvent this limitation. The new model became known as Generalising Random Access Memories (GRAM)[164] and its main feature is its spreading phase, which creates clusters, whose centroids are the learned patterns. This phase can have as many iterations as previously defined and in an iteration i only the u-positions that are a Hamming distance i from the pattern will learn it. Due to a large amount of memory required when the number of clusters increases, an alternative implementation of GRAM was created. Known as Virtual GRAM (VGRAM)[165], it only instantiates memory locations that have been trained, so that the network generalizes patterns not seen in training through calculations using the stored values and the spreading algorithm.

GNU: General Neural Unit (GNU)[166] are single-layer networks where each neuron is a GRAM, each of which has n inputs connected to the input layer and m nodes connected to the output. They can be trained as feedforward or/and feedback networks. The process of storing this model consists of creating associations between an external pattern and its representation. In the training phase, the same GRAM algorithm is applied to each neuron. In this phase the learned pattern is associated with itself, what is known as *iconic learning*.

Subsequently, a system was created by combining several GNUs for modeling brain structures. He became known as MAGNUS and the experiments carried out with him were one of the drivers for the discussion on the possibility of Artificial Consciousness[167].

ADAM[168] is a network whose architecture uses two weightless correlation matrix memories using a Hebbian learning rule[169]. **CAINN[170]** is an adaptation of the ADAM architecture using Alpha-Beta operations.

SDM[171] is a generalized RAM inspired by the concept of long-term memory. It uses sensitivity between two memory locations so that a word of data can be retrieved not only if its address is accessed, but also some other similar memory location is accessed. The similarity between memory locations is calculated by the number of mismatched bits (i.e., the Hamming distance between memory addresses).

Fuzzy Boolean Neural Network[172] is a model whose memory locations can store "0", "1" or "u" and which are capable of learning qualitative rules and of reasoning using those rules.

Quantum weightless neural networks[173] use qRAM neurons, whose memory locations store qubits. These do not use non-linear networks activation function, like sigmoid or tangent hyperbolic, because non-linear activation functions will hardly have an exact quantum analogous. It is possible to simulate using the classic WANNs learning algorithms using qRAMs.

2.7 WAAN's Recent Advances

After the addition of *bleaching* and its consequent reduction in model saturation, WiSARD was once again competitive in tasks from the most diverse domains, many of which are real world problems.. The main ones will be listed below:

- **Online tracking:** this is a complex task since characteristics of the environment can change in real-time, as well as the object being tracked can suffer occlusions and deformations. WiSARD was used effectively in this task in an approach inspired by the hierarchy of human memory.

  In this domain, only a single discriminator is created and the model has no changes in its learning phase. In the classification phase, the entire scenario is covered by a sliding window looking for which specimen will have the highest score. If the score of the classified object does not reach a pattern threshold, a new discriminator will be created to learn this new configuration of the object, which must have changed from its original shape.

  This system has two queues, corresponding to short and medium-term memories. When a new discriminator is created, it occupies a position in short-term memory, and when a discriminator fails to reach the pattern threshold it is moved to medium-term memory. Both memories have a size limit and when this is reached the last discriminator is removed from the system. Results of this system in online video tracking can be seen in [174];

- **GPS trajectory classification**, a complex task since GPS data usually involves a lot of noise. A successful approach using WiSARD in this domain consists of preprocessing the data using kd-tree division of the space[175] and using an ensemble of WiSARDs arranged in a decision directed acyclic graph[176], forming a decision tree-like structure, where spurious data is eliminated and ties are avoided through a neurosymbolic methodology. Results are found in [177, 178];

- **Part-of-speech (PoS) Tagging** is a complex task due to the presence of homonyms and the great vastness of words in most languages, there is a high chance that there would be many words that were not present in the training set. An efficient multilingual PoS-tagging was built based on the WiSARD classifier. Since this system has many parameters to be tuned depending, and that are very sensitive to specific characteristics of each language, such as its synthesis index, the use of a fast and light model like WiSARD allows an exhaustive search for the best configuration of each language. The system, which became known as mWANN-Tagger, has in addition to a WiSARD classifier, a mapping matrix that relates the degree of relevance of the words in the corpus and a context window, which aims to reduce homonymy. There are more results from mWANN-Tagger in eight different languages[179–181].

Although WANNs are not ML mainstream models, their use has always remained in certain AI niches, due to their versatility and low computational cost. To date, attempts to circumvent the limitations of these models persist, largely stemming from the difficulty of consistently representing complex data with binary data words. Its use is mainly associated with hardware implementations and environments where little memory is available or online learning is required.

Among the recent work with WANNs, in addition to those already mentioned in the Section 2.6, is the use of AutoWiSARD as a basis for reinforcement learning techniques[182–184], use of WiSARD in a feasible weightless neural accelerator for a fairly small-sized Xilinx FGPA[70], use of WiSARD as an accelerator of learning in deep neural networks[62], use of WiSARD with transfer learning to detect distress in asphalt[63], generation of synthetic datasets[185], cryptography[186], combined use of VGRAM with a modified binary TRIE data structure and genetic algorithms for automatic disease diagnosis[187] and autonomous vehicle control systems based on WiSARD[188]. A recent survey has discussed the relevance of n-tuple classifier in a contemporary ML scenario[106].

2.8 Preprocessing Techniques

As mentioned in Chapter Two, the input of WiSARD is conveniently composed of bits. This way, it was necessary to apply some preprocessing to the system entry, so that it can be represented as a collection of Boolean values.

2.8.1 Image

In the case of images, the most trivial preprocessing procedure consists of binarizing every pixel, as a means to highlight some particular type of the desired feature in the image, such as shapes, borders, or clusters with shared characteristics. The image preprocessing techniques employed in this work were Sauvola, Canny edge detector, adaptive Gaussian, and Otsu binarization.

The simplest binarization for images consists of establishing a global threshold based on some graphical feature, such as the average intensity of their pixels, and assigning a value for each pixel according to this threshold. However, assigning the same threshold for the whole image produces noises in the case of non-uniform backgrounds. Because of this, we employed techniques based on local information, and the feature used to calculate the threshold was luminance.

2.8.1.1 Deskewing

When we normally write something, there is an angle with the paper. In the image recognition process, this ends up being a type of noise. To minimize this, a process known as deskewing is used, which consists of an affine transformation.

For this, is assumed that the image that was created (the skewed version), it is actually some affine skew transformation on the image $Image' = A(Image) + b$, that is unknown. Therefore, one must calculate the matrix that allows obtaining this image. To do this, it is necessary to calculate the covariance of the pixel intensity of the image, since the matrix that skew back to original image is: $\begin{bmatrix} 1 & 0 \\ \alpha & 1 \end{bmatrix}$, where $\alpha = \frac{Cov(X,Y)}{Var(X)}$. Then, this matrix and the offset μ - center are used to perform interpolation and obtain the deskewed image.

2.8.1.2 Yen's Binarization

Yen's binarizartion[189] is multilevel thresholding. This method uses two factors to determine threshold values: the discrepancy between the thresholded and original images and the number of bits required to represent the thresholded image. Based on a new maximum correlation criterion for bilevel thresholding, the discrepancy is defined and then a cost function is proposed for multilevel thresholding. By minimizing the cost function, the classification number that the gray-levels should be classified and the threshold values can be determined automatically.

2.8.1.3 Adaptive Gaussian

Adaptive Gaussian is a low-pass filter that binarizes the image according to a local threshold for each pixel, defined by the convolution of the neighboring pixel features weighted with a Gaussian kernel.

2.8.1.4 Sauvola's Binarization

Sauvola method [190], in turn, makes use of the concept of the integral image, i.e., a version of the image in which every pixel has its value substituted by the sum of the values of every pixel that lie to left and/or above it. Given that integral image, Sauvola binarizes a pixel through the application of a threshold t defined by

$$t = 1 + \mu_F + k\left(\frac{\sigma_F}{r} - 1\right), \tag{2.6}$$

where F represents the features of the neighboring pixels, μ_F and σ_F their respective mean and standard deviation, and k and r are factors responsible for amplifying the contribution of s_F in an adaptive manner in order to minimize potential harming effects caused by the background.

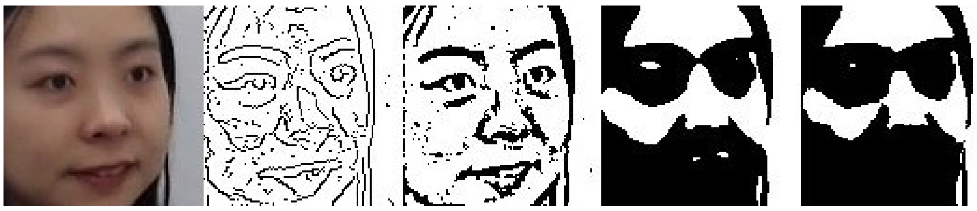


Figure 2.4: Preprocessings: (a) original image[3]; (b) Canny filter; (c) Adaptive Gaussian filter; (d) Sauvola binarization; (e) Otsu binarization.

2.8.1.5 Canny Border Detector

The Canny border detector is a technique widely used in computer vision to reduce the amount of information to be processed. This method smooths the image by applying it to a Gaussian filter, then calculates its intensity gradients and removes spurious edges using non-maximum suppression techniques. It then applies two empirically determined thresholds and classifies the pixels according to their gradients. A pixel whose gradient is greater than the highest threshold is classified as strong, if its gradient lies between the thresholds it is classified as weak, and it is discarded if the gradient is lower than the lowest one. Next, weak pixels that are not bound to strong ones are also suppressed.

2.8.1.6 Otsu's Binarization

Otsu's binarization employs a clustering approach. This method is applied only to grayscale images and considers that there are only two types of pixels based on a bimodal histogram. The algorithm calculates an optimal threshold that separates these pixels through the minimization of intraclass variance $\sigma^2_{\text{intra}}(t)$, which is given by

$$\sigma^2_{\text{intra}}(t) = p_S(t)\,\sigma^2_S(t) + p_W(t)\,\sigma^2_W(t)\,, \tag{2.7}$$

where $\sigma^2_S(t)$ and $\sigma^2_W(t)$ are the variances of the gradients of strong and weak pixels, and $p_S(t)$ and $p_W(t)$ are the probabilities of a pixel being of that class given threshold t. Otsu's method is based on the fact that minimizing intraclass variance $\sigma^2_{\text{intra}}(t)$ is the same as maximizing interclass variance $\sigma^2_{\text{inter}}(t)$, which is given by

$$\sigma^2_{\text{inter}}(t) = p_S(t)\,p_W(t)\,(\mu_S(t) - \mu_W(t))^2, \tag{2.8}$$

where $\mu_S(t)$ and $\mu_W(t)$ are the means of the gradients of strong and weak pixels given threshold t.

A comparison of the binarized images produced by the different preprocessing methods is depicted in Figure 2.4.

2.8.2 Audio

Audio processing, in turn, has the goal of making the entry compatible with the canvas employed to treat the time series and also to highlight features of interest to the network. This preprocessing procedure consists of performing an MFCC (mel frequency cepstral coefficients) feature extraction, which produces acoustic feature vectors from the original waveform, and then in the binarization of those vectors with KernelCanvas. The detailed steps of the procedure of converting the audio into binary inputs are the following:

- application a pre-emphasis filter on the signal to increase the amplitude of high-frequency bands and decrease the amplitude of low ones. The pre-emphasis filter is a first-order high-pass filter that if applied to a signal $x(t)$ produces accentuated signal
$$y(t) = x(t) - \alpha x(t-1)\,, \tag{2.9}$$
for $0.9 \leq \alpha \leq 1.0$;

- windowing of the audio signal, by sampling extracts of fixed length from the original audio in frames. This sampling is performed according to the frame size (in milliseconds) and the frameshift, which is the offset between successive frames. It is important that the frame size be greater than the frame shift, so that there is an overlap between the frames. Solely framing the original waveform produces signals that are abruptly cut off at their boundaries and these discontinuities create complications in the upcoming Fourier analysis. To ensure continuity, a Hamming window is applied to the framed signal so that the amplitudes at the boundaries are shrunk to 0. The Hamming window for a frame of length N is given by the formula
$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right), \tag{2.10}$$
where $0 \leq n \leq N-1$;

- conversion of the Hamming-windowed signal to a frequency spectrum through the use of the discrete Fourier transform (DFT). A much more efficient algorithm called fast Fourier transform (FFT) can be employed in this step, but it only works for frames of length N that are powers of 2;

- elimination of extremely high frequencies, e.g. 16 kHz, and creation of a bank of high-pass filters for the remaining frequencies. The human auditory system is less sensitive in higher frequencies than it is in lower ones, and this bank must try to replicate this property. So, its filters are made with overlapping triangular windows whose centers are spaced almost linearly below 1000 Hz and logarithmically above it. More specifically, the centers of those filters are linearly spaced in the mel scale, which is given by

$$\mathrm{mel}(f) = 1127 \ln\left(1 + \frac{f}{700}\right), \tag{2.11}$$

where f is a frequency in hertz. Lastly, the amplitudes of the filters are subjected to a logarithm for human perception of amplitude is logarithmic;

- the high correlation between the filter banks is problematic in some models of machine learning, then a discrete cosine transform (DCT) should be applied to decorrelate them and reduce their representation. The results of this transform are known as the mel frequency cepstral coefficients, the components of an audio representation obtained by the spectrum of the log of the spectrum of the time signal;
- to balance the spectrum and improve the signal-to-noise, the average of each coefficient of all frames is subtracted;
- to make better use of the context of the audio window, an integral image of the input is constructed;
- the resulting input is then binarized with KernelCanvas.

When Gaussian mixture models were the main machine learning solution for audio recognition, MFCC was the main type of representation used because of the limitation that many machine learning algorithms had to handle highly nonlinear data. With the rise of deep neural networks, this was no longer a problem, and the use of filter banks alone became more popular as this reduces the cost of data processing and preserves sample information. Here only MFCC was used for binary input composition.

2.8.3 Thermometer

Binary thermometer is a technique for preprocessing quantitative variables. Given a variable d, a maximum value of training test m and a number of ranges s, the new binary variable will have s bits, with each ith bit being determined by a threshold $t = i * \frac{m}{s}$. If $d > t$, the ith position is worth 1, otherwise 0. A example is displayed in Figure 2.5, where $d = 76, s = 7, max = 90$. Some treatment should be given to values in test set that are greater than the maximum value of the training set. The main advantage in use thermometer in relation to represent a number using the direct binary representation: thermometer preserves Hamming distance between

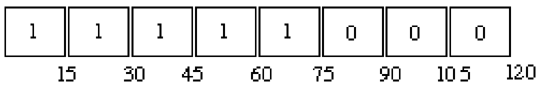


Figure 2.5: Thermometer encoding example

the binary inputs. Another advantage of the thermometer is that it can scale the representation of a feature so that its impact on classification is not dissipated in front of other features. A total comparison between both of them in different scenarios in order to prove in which they are advantageous and disadvantageous is still needed.

2.8.4 One-hot-encoding

Binary preprocessing technique for categorical variables, where the number of bits of the variable is fixed as the number of possible values for the variable in the training set. Each of these values will correspond to a binary variable setting, where only one position has "1". Table 2.1 display a example of this encoding. Some treatment should be given to test set values that are not found in the training set.

Table 2.1: Example of one-hot-encoding for variable Fantastic Creature = [griffin, elf, dragon, fairy

Variable	**Encoding**
griffin	1000
elf	0100
dragon	0010
fairy	0001

2.8.5 tf-idf

A numerical statistic that is intended to reflect how important a word is to a document in a corpus. It is formed by the product between term frequency[191], which is the frequency of a word in a given document in the corpus, and the inverse document frequency[192], which is the proportion of occurrences of the same word in documents in the corpus, being a fixed value for the entire corpus.

Table 2.2: Term count in Document 1

Term	**Term count**
Wisdom	1
is	7
light	4

Table 2.3: Term count in Document 2

Term	**Term count**
Ignorance	2
is	5
shadow	1

Considering a corpus $D = D1, D2$ and the frequency of some of its terms listed in Tables 3.2 and 3.3, the tf-idf of the word is calculated:

- "is" in Document 1 $= \frac{7}{12} * log(\frac{2}{2}) = 0.583 * 0 = 0$
- "is" in Document 2 $= \frac{5}{8} * log(\frac{2}{2}) = 0.625 * 0 = 0$
- "light" in Document 1 $= \frac{4}{12} * log(\frac{2}{1}) = 0.33 * 0.301 = 0.099$

Analyzing the tf-idf features of these words, it can be concluded that the word "is" is not important in the classification of elements in this corpus given its high incidence in different contexts.

2.8.6 Discussion about the Preprocessing Techniques

The efficiency of such preprocessing techniques depends on each domain and on the network topology, so it is not possible to generalize its performance in general. In the following chapters we will have analysis of these preprocessing in 11 different datasets. Other analyzes of the use of these preprocessing techniques with the WiSARD model were made in literature[2, 116, 125, 131, 193–195].

Chapter 3

WiSARD in Action Units Multi-label Classification

Many ML classification tasks involve multi-label classification, which implies not in the categorization of samples, but in the selection of a subset of domain labels for these, and in label ranking, which consists of ordering the labels of a given domain to the degree of belonging to a given example. Label ranking is a task that can be easily solved by a WiSARD classifier since with the score obtained by each discriminator in the classification stage, such ordering becomes a trivial task. However, like many other classifiers, WiSARD is unable to select a subset of labels without an domain-oriented modeling. In this chapter, we will explore the use of WiSARD with some methodologies to expand its traditional use and solve a multi-label classification task: the classification of Action Units.

Ekman and Friesen [196] cataloged a set of muscles known as *Action Units* (AUs), which would be responsible for all facial expressiveness while attempting to obtain a set of universal emotions present in any human. The automatic identification of these AUs has been developed since the mid-1990s and has several applications: forensics, psychological treatment, physical therapy support, and advertising feedback, among others. AUs have also been used in the development of adaptive digital avatars [197].

Some of the great difficulties in automatic detection of AUs are a large number of classes and the wide variety of forms how AUs express themselves, besides the fact that they usually manifest together, making this a hard multi-label task. In this way, the approaches that are emerging in the literature usually involve complex techniques of computer vision and machine learning, with high computational cost and long learning time. This work presents an alternative that solves that problem by using weightless neural networks (WANNs), which are characterized by their simplicity of implementation and online learning. The weightless solutions presented here are validated in the CK + dataset, which presents 30 classes of AUs.

This chapter is organized as follows: Section 2 introduces the two approaches

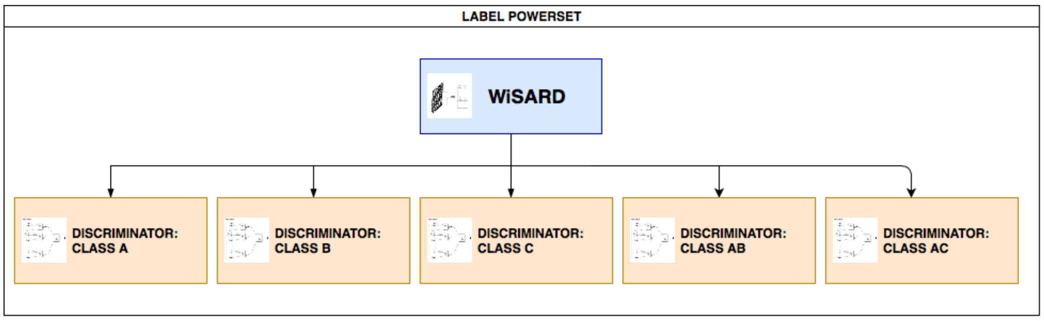


Figure 3.1: Label Powerset

used in order to reduce the multi-label problem classification of AUs in acceptable single-label problems for WANNs. Section 3 presents the concept of AUs and their many classifications. In Section 4, the experiments are performed using supervised and semi-supervised datasets. Although a description of state-of-the-art works and other relevant solutions with their main characteristics is provided, unfortunately, a complete comparison with them is not possible due to lack of information on which classes are used by them and their performance in relation to training time and test. Section 5 concludes the text, highlighting the contribution of this work in providing multi-label solutions for the weightless neural models while offering a simpler and faster solution for the AU classification.

3.1 WiSARD-based Multi-label Classification Systems

Since WiSARD uses a set of discriminators to infer a single class that a given input is more likely to belong to, two traditional multi-label strategies[198] have been adapted to work with this paradigm. They are Label Powerset and Binary Relevance.

3.1.1 Label Powerset

In this approach a combination of classes is considered as a new class. When the number of classes increases much in relation to the single-label problem, and to circumvent memory spending, new discriminators are instantiated when a new class is presented in the training phase. This approach is illustrated in Figure 3.1.

One problem with this solution is that a misclassified AU will induce other erroneous classifications since the network can only find a single group of AUs. In this approach, when the amount of discriminators increases, the amount of training examples will tend to decrease and this can consume a lot of memory depending on the combination of classes present in the training dataset.

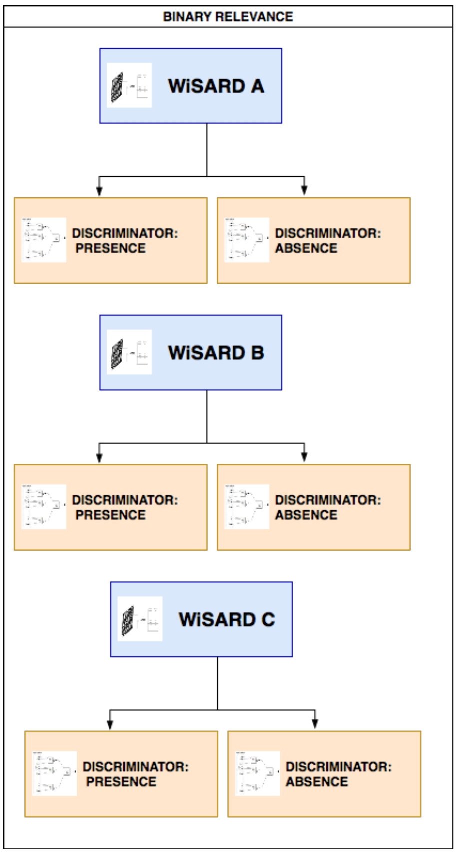


Figure 3.2: Binary Relevance

3.1.2 Binary Relevance

The idea is to use a set of WiSARDs where each one is related to an AU, all with two discriminators indicating the presence or the absence of AU. In the training phase, when an example is submitted, all the WiSARDs are trained in the appropriate discriminator. In the classification phase, AUs activation will be predicted according to the response of each of the WiSARDs. This approach is illustrated in Figure 3.2.

The disadvantage of this method over the Label Powerset is the fact that, since the combination of non-additive AUs is extremely idiosyncratic, it may not be captured by the WiSARDs responsible for its elements. On other hand, this method has the advantages of making classification of Upper and Lower Face AUs independents and save memory and classification time in relation to Label Powerset. In this approach is possible that no class appears in the output.

3.2 Action Units

The study of human expressions has been developed through scientific analysis since the second half of the 19th century but, due to the lack of objectivity in defining parameters to categorize such expressions, this research was unable to be properly developed until the end of the 1970s, when Ekman and Friesen finally proposed the use of facial muscles as the physiological element, totally independent of cultural context, to support a taxonomy of the face. From there they developed the Facial Action Coding System (FACS) [196], a descriptive code of facial expressions, whose basic units are the 32 muscles that express facial emotions and 14 Action Descriptors (ADs), which describe facial actions not expressed through muscles, such as the movement of the eyes, tongue, and jaw.

AUs are divided into Upper Face (above the nose) and Lower Face (from nose to chin) and into additive (their appearance is independent of the rest of the face) and non-additive (their appearance may be affected by other non-additive AUs, which are in the same facial region). The 32 AUs can form more than 7000 combinations[199]. AUs may manifest voluntarily or involuntarily, which affects the duration of their manifestation and their symmetry. Involuntary AUs are used to detect micro-expressions. AUs have different degrees of intensity, usually divided into five levels of expressiveness.

3.3 Related Work

A lot of work has been developed for the detection and classification of Action Units, since this is useful in a wide range of tasks, such as medical, pedagogical, and computer graphics applications. Most of the systems found in the literature use some computer vision technique to extract features, which are then fed to a classifier. Among the techniques for extracting features, some popular solutions are Optical Flow[200] (movement vector of objects per frame), Gabor Wavelets[201] (transform capable of obtaining spatio-temporal information from a signal, being able to fully represent images), Multi-state Models[202] (they model a process with several transitions, where each state can be submitted to a distinct heuristic), KLT Tracker[203] (information-using method about the spatial intensity of the image to reduce the computational cost of searching for specific features) and PBVD Tracker[204] (similar to the previous one, it uses the Bézier partial volume deformation model).

The classifiers with the best results are Bayesian networks[205] (knowledge representation models that work with uncertain and incomplete knowledge through Bayes' Theorem), Hidden Markov Model[206] (the statistical model in which the modeled system is assumed as a Markov process with unknown parameters), SVM[207] (supervised learning model that separates distinct classes through a hyper-plane) and neural networks.

Some specific solutions for AU classification: Khorrami et al.[208] train a zero-bias CNN on facial expression data and use an approach to decipher which portions of the face influence the CNN's predictions. This work also uses the FAU labels provided in the CK+ dataset to verify that the FAUs observed in their filter visualizations indeed align with the subject's facial movements.

Prajod et al.[209] train a VGG16 convolutional neural network to discern between emotions, then fine-tune larger parts of this network to learn suitable representations for pain recognition. Then use Layer-wise Relevance Propagation to analyze predictions of the model that have been predicted correctly previously but are now wrongly classified. Chang et al.[210] use the rough contour estimation routine, mathematical morphology, and point contour detection method to extract the precise contours of the eyebrows, eyes, and mouth of a face image.

In the literature[211][212][213] other relevant results of classification of AUs in CK+ do not use all AUs or use other techniques to increase the available information. Breuer & Kimmel[214] uses transfer-learning and all-against-all validation, with only 8 AUs, obtaining an accuracy of 98.62%. Abbasnejad et al.[215] uses a synthetic dataset to extend the amount of data and 11 AUs are used and accuracy is 97.87%. Pons & Masip[216] validates with 12 AUs and maximum accuracy is 82.5%. There is no indication of which metric was used to calculate the accuracy in these works.

The main benchmark found in the literature in the CK+ dataset[4] for AU classification uses leave-one-out as validation. The state-of-the-art uses a pair-wise classifier trained with time series[212], in order to use the facial changes in each frame to highlight the present AUs. The baseline proposed in [4] uses Active Appearance Models and a linear support vector machine classifier. This comparison is better visualized in Table 3.1.

3.4 Experimental Results

3.4.1 Experimental Setup

The Extended Cohn-Kanade dataset[4] was chosen for the experiments. It consists of a base created by the University of Pittsburgh Affective Analysis Group. It is made up of 500 image sequences from 100 subjects, aged between 18 and 30 years. 65% of individuals are female. 82% of the individuals are Caucasian, 15% African American and 3% Latino and Asian. All images are completely frontal and are all posed.

Each series of images on this base is made up of 23 photographs, each of which has at least one Action Unit, the muscle used to express emotion, prominently active, and can exist combinations of several of them in a single image. The first image in each series exhibits neutral or predominantly neutral emotion, and throughout the images, some other emotion becomes predominant until it reaches the apex of its expressiveness. Each image is annotated with information about the emotion

Table 3.1: A comparison between ML-solutions for AUs classification in literature.

Works	**Model**	**Other techniques**	**Validation**	**Subset**
[208]	Zero-bias CNN	FAU-oriented Filter	10-fold CV	First frame of each sequence as a neutral frame in addition to the last three expressive frames
[209]	VGG16 CNN	Layer-wise Relevance Propagation	Train-test split	The images were collected through search queries containing emotional keywords; imbalanced
[210]	Radial basis function network and MLP	Rough contour estimation routine, mathematical morphology, and point contour detection	Train-test split	All data
[214]	DNN	Transfer-learning	All-against-all	8 Most frequent AUs
[215]	DNN	Data augmentation	10-fold CV	11 Most frequent AUs
[216]	DNN	-	10-fold CV	12 Most frequent AUs
[4]	Linear SVM	Active Appearance Models	Leave-one-out	All data
[212]	Pair-wise Multi-label Perceptron	Temporal information	Leave-one-out	Most frequent AUs

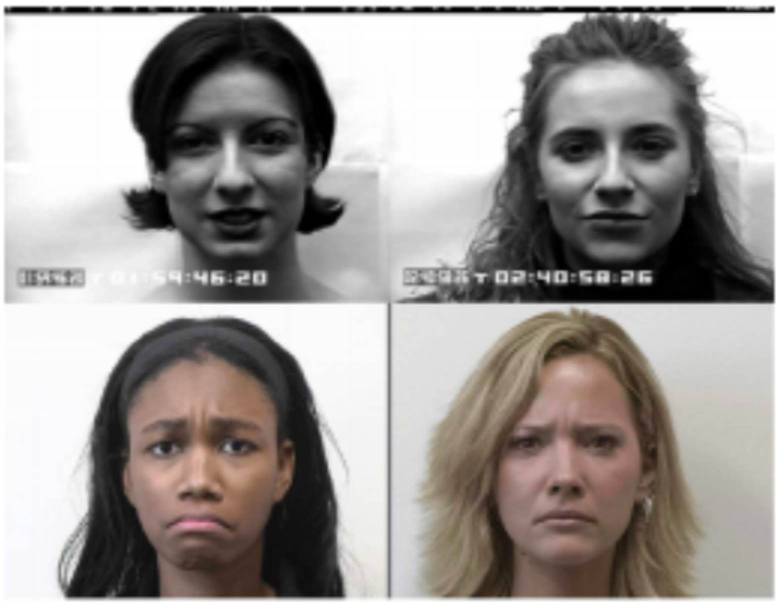


Figure 3.3: Examples of Cohn-Kanade Extended Dataset (CK+)[4]

displayed and the AUs expressed in it. Figure 3.3 gives examples of the images in this dataset:

It is worth mentioning that of 10558 images present in the dataset, only 588 have annotations indicating which AUs, and with what intensity, are present.

Table 3.2 presents the description of the AUs present in CK+, with their frequency. One of the great difficulties of this dataset can be perceived here, by the imbalance of the classes. Another difficulty is due to the fact that additive AUs modify completely when they mutually manifest so that the combination of AUs can practically be considered new classes (with many different features of the classes that compose it).

This dataset was chosen to validate multi-label classification systems with WiSARD due to its recurrent use in the literature, WiSARD's natural vocation for image classification (both n-tuple classifier and WiSARD were created with this intention and most of them of the WiSARD application still focuses on this area) and because WiSARD has already been applied to it, but to the task of classifying emotions[194].

The preprocessing methods used were Adaptive Mean, Adaptive Gaussian and, Sauvola binarization[190]. The first two techniques binarize the image according to a local threshold for each pixel defined by the mean luminance of the neighborhood and the weighted-sum of Gaussian window, respectively, while the Sauvola method uses integral images for computation of the threshold.

All experiments were run three times. The experiments were run on a machine with the following configuration: 7.7 GiB, Intel Core i7-6500U CPU @ 2.50GHz x 4, GeForce 920MX/ PCIe/ SSE2, 64-bit, Ubuntu 18.04.1.

Table 3.2: Number of examples of each AU in CK+ dataset

AU	Name	N	AU	Name	N
1	Inner Brow Raiser	173	18	Lip Puckerer	9
2	Outer Brow Raiser	116	20	Lip Stretcher	77
4	Brow Lowerer	191	21	Neck Tightener	3
5	Upper Lip Raiser	102	23	Lip Tightener	59
6	Cheek Raiser	122	24	Lip Pressor	57
7	Lid Tightener	119	25	Lips Part	287
9	Nose Wrinkler	74	26	Jaw Drop	48
10	Upper Lip Raiser	21	27	Mouth Stretch	81
11	Nasolabial Deepener	33	28	Lip Suck	1
12	Lip Corner Puller	111	29	Jaw Thrust	1
13	Cheek Puller	2	31	Jaw Clencher	3
14	Dimpler	29	34	Cheek Puff	1
15	Lip Corner Depressor	89	38	Nostril Dilator	29
16	Lower Lip Depressor	24	39	Nostril Compressor	16
17	Chin Raiser	196	43	Eyes Closed	9

3.4.2 Cross-validation with Full Dataset

Both networks (WiSARD and ClusWiSARD) were tested in combination with both methods (Binary Relevance and Label Powerset) using only annotated CK+ images in 10-fold cross-validation. The landmarks information provided in the dataset was used to obtain the box used to generate the input from the network.

In these experiments, the accuracy was calculated as: $acc = \frac{tp+tn}{tp+tn+fp+fn}$, that is, to calculate the accuracy, the system's errors and absolute hits were considered and not the number of class hits for each sample. Other metrics considered here were F1-score, recall and precision. Since there is an imbalance between classes, F1-score is a better metric than accuracy in this task.

Tables 3.2-3.6 show the best accuracy, F1-score, precision, recall and ROC AU score, respectively, for each network type combination, multi-label approach, and preprocessing. More results are available in Appendix A.

In relationship to accuracy, Adaptive Mean and Adaptive Gaussian preprocessing methods worked better with Binary Relevance, while Sauvola provided better results with Label Powerset. The best result obtained in this sense was 89.66%, using ClusWiSARD with 6 bits of addressing, Binary Relevance approach, and Adaptive Mean preprocessing. Since there is no significant difference between these accuracy values, these results are insufficient to determine which of the approaches was superior in terms of accuracy, and statistical tests that consider the null hypothesis are necessary to obtain a truly accurate assessment.

We can see from the Tables 3.4 and 3.5 that Binary Relevance performs well on Recall but underperforms on Precision, indicating that this method is better at

Table 3.3: Best accuracy results for each combination of type of network, multi-label approach and preprocessing.

Network	**Multi-label**	**Preprocessing**	**Acc (%)**
WiSARD	Binary Relevance	Sauvola	83.94
		Adaptive Mean	89.54
		Adaptive Gaussian	89.54
	Label Powerset	Sauvola	87.8
		Adaptive Mean	86.03
		Adaptive Gaussian	86.88
ClusWiSARD	**Binary Relevance**	Sauvola	85.67
		Adaptive Mean	**89.66**
		Adaptive Gaussian	89.54
	Label Powerset	Sauvola	87.85
		Adaptive Mean	86.21
		Adaptive Gaussian	87.02

Table 3.4: Best F1-score results for each combination of type of network, multi-label approach and preprocessing.

Network	**Multi-label**	**Preprocessing**	**F1-score**
WiSARD	Binary Relevance	Sauvola	38.1 ± 0.04
		Adaptive Mean	36.9 ± 0.011
		Adaptive Gaussian	38.3 ± 0.017
	Label Powerset	Sauvola	46.1 ± 0.037
		Adaptive Mean	41.5 ± 0.041
		Adaptive Gaussian	41.3 ± 0.034
ClusWiSARD	Binary Relevance	Sauvola	35.4 ± 0.017
		Adaptive Mean	34.9 ± 0.021
		Adaptive Gaussian	49.11 ± 0.034
	Label Powerset	**Sauvola**	**49.11 ± 0.011**
		Adaptive Mean	42.1 ± 0.011
		Adaptive Gaussian	40.7 ± 0.014

Table 3.5: Best precision results for each combination of type of network, multi-label approach and preprocessing.

Network	**Multi-label**	**Preprocessing**	**Precision**
WiSARD	Binary Relevance	Sauvola	31.1 ± 0.037
		Adaptive Mean	52.1 ± 0.011
		Adaptive Gaussian	52.1 ± 0.024
	Label Powerset	Sauvola	58.7 ± 0.027
		Adaptive Mean	49.1 ± 0.014
		Adaptive Gaussian	44.1 ± 0.031
ClusWiSARD	Binary Relevance	Sauvola	35.1 ± 0.047
		Adaptive Mean	53.7 ± 0.021
		Adaptive Gaussian	55.1 ± 0.011
	Label Powerset	**Sauvola**	**59.7 ± 0.041**
		Adaptive Mean	48.1 ± 0.034
		Adaptive Gaussian	42.7 ± 0.047

Table 3.6: Best recall results for each combination of type of network, multi-label approach and preprocessing.

Network	**Multi-label**	**Preprocessing**	**Recall**
WiSARD	**Binary Relevance**	Sauvola	91.1 ± 0.02
		Adaptive Mean	**99.7 ± 0.011**
		Adaptive Gaussian	**99.7 ± 0.014**
	Label Powerset	Sauvola	41.7 ± 0.01
		Adaptive Mean	41.4 ± 0.01
		Adaptive Gaussian	41.1 ± 0.014
ClusWiSARD	**Binary Relevance**	Sauvola	91.1 ± 0.017
		Adaptive Mean	**99.7 ± 0.014**
		Adaptive Gaussian	**99.7 ± 0.011**
	Label Powerset	Sauvola	43.1 ± 0.02
		Adaptive Mean	41.1 ± 0.031
		Adaptive Gaussian	39.7 ± 0.01

Table 3.7: Best ROC AUC score results for each combination of type of network, multi-label approach and preprocessing.

Network	**Multi-label**	**Preprocessing**	**ROC AUC**
WiSARD	Binary Relevance	Sauvola	0.485 ± 0.016
		Adaptive Mean	0.466 ± 0.037
		Adaptive Gaussian	0.466 ± 0.037
	Label Powerset	**Sauvola**	**0.541 ± 0.005**
		Adaptive Mean	0.521 ± 0.013
		Adaptive Gaussian	0.51 ± 0.015
ClusWiSARD	Binary Relevance	Sauvola	0.481 ± 0.017
		Adaptive Mean	0.465 ± 0.036
		Adaptive Gaussian	0.462 ± 0.037
	Label Powerset	Sauvola	0.539 ± 0.007
		Adaptive Mean	0.517 ± 0.011
		Adaptive Gaussian	0.509 ± 0.008

Table 3.8: The training and classification time results for the best combination of type of network, multi-label approach and preprocessing in relation to F1-score; TrT - training time; TT - test time.

Network	**Multi-label**	**Preprocessing**	**TrT (s)**	**TT (s)**
WiSARD	Binary Relevance	Sauvola	0.24	0.0005
		Adaptive Mean	0.24	0.0005
		Adaptive Gaussian	0.25	0.0005
	Label Powerset	Sauvola	0.0006	0.0045
		Adaptive Mean	0.0006	0.0045
		Adaptive Gaussian	0.0006	0.0045
ClusWiSARD	Binary Relevance	Sauvola	0.29	0.0006
		Adaptive Mean	0.30	0.0006
		Adaptive Gaussian	0.30	0.0006
	Label Powerset	Sauvola	0.00065	0.007
		Adaptive Mean	0.00065	0.007
		Adaptive Gaussian	0.00065	0.007

avoiding false negatives, but performs worse when false positives receive a higher penalty. This may indicate that in this method many WiSARDs were classifying the samples as belonging to a particular AU, so that the true classes of the sample were identified by the excess of positive votes, which degraded performance in Precision. Label Powerset was balanced in Recall and Precision, having better performance in Precision than Binary Relevance and having worse performance concerning Recall. This indicates that this system was more conservative than Binary Relevance, having a lower false positives rate and a higher false negatives rate.

Another important analysis in multi-label classification is ROC AUC score, that to refers to the area under a ROC curve, responsible to diagnose the classifier system as its discrimination threshold is varied. The results obtained here are shown in Table 3.6 and ClusWiSARD in Label Powerset using Sauvola preprocessing obtained the best result. Actually, Label Powerset performed better regardless of preprocessing, thus indicating the ability of this method to distinguish the positive class values from the negative class values better than Binary Relevance. Using this metric, WiSARD performed better than ClusWiSARD.

Since the dataset is unbalanced, accuracy may not be a fair metric, as it takes advantage of correct classifications of more frequent classes. Using AUC ROC score, only systems with Label Powerset obtained a score greater than 0.5, indicating that only these systems had a considerable success rate, that is, they were superior to a random guess. However, [217] discusses how AUC ROC should not be used on unbalanced datasets, as the number of false positives will be reduced due to the high number of true negatives, which is particularly true in this dataset, where there are dozens of classes and only a few label per sample.

Table 3.7 show the training and classification time results for the best combination of type of network, multi-label approach and preprocessing in relation to F1-score. In terms of training time, Label Powerset is faster, independent of the network, because it needs to train in only one discriminator per input. In terms of classification speed, Binary Relevance with WiSARD was more efficient due to the lower number of classes than Label Powerset and due to the smaller number of discriminators compared to ClusWiSARD.

Smaller addressing values had better F1-Score and precision results with Binary Relevance and recall with Label Powerset, while higher bit-addressing values resulted in better F1-Score and precision scores with Label Powerset and recall with Binary Relevance, besides better training and classification time, regardless of method.

3.4.3 Cross-validation with Subsets

Given the low occurrence of some AUs and their combinations, so unique if compared to their combinations, two tests were done using subsets of AUs. A test was performed with ClusWiSARD, $n = 6$, by removing all AUs that do not have sufficient examples in the dataset, reducing to 324 images for the AUs 1, 2, 4, 5, 6, 7, 9, 12, 15, 17, 20, 25 and 27, with 33.33% of perfect matches between classifications and labels, F1-score of 61.46%, precision of 67.13% and recall of 56.67%. The accuracy obtained was 83.02%. With the same configuration of ClusWiSARD: 11 AUs - acc = 86.77%, F1-score = 74.09% (removing AUs 9 and 20 of previous subset); 9 AUs (removing AUs 15 and 17) - acc = 85.71%, F1-score = 76.65%. This represented a significant improvement in the F1-Score and the number of perfect matches, but a slight drop in accuracy.

In order to increase the classification quality, another test was performed using all annotated images and 4242 images without annotation in a ClusWiSARD with $n = 25$, minimum score = 0.1, growth interval = 10, and three maximum discriminators per class, in a Label Powerset approach with Sauvola preprocessing, obtaining accuracy = 32.8% and F1-score = 34%, representing a drop in performance on both attributes.

3.4.4 Leave-one-out Validation

WiSARD is compared with state-of-the-art and baseline, in which the same classifier is used, but without the context window. Both of them are more elaborated in Section **??**. The best result for weightless systems here was WiSARD using Label Powerset and Adaptive Mean and the F1-score was the metric used in this comparison, both for each AU, as for the entire dataset. The benchmark is showed in Table 3.8.

Table 3.9: Benchmark using F1-score in leave-one-out validation

AUs	**WiSARD**	**Baseline**	**Relative Facial**
AU 1	0.644	0.94	**0.95**
AU 2	0.79	**0.97**	**0.97**
AU 5	0.74	0.95	**0.97**
AU 6	0.73	0.92	**0.94**
AU 9	0.8	**0.98**	**0.98**
AU 12	0.74	0.91	**0.93**
AU 15	0.38	0.80	**0.83**
AU 17	0.39	0.84	**0.86**
AU 20	0.17	0.91	**0.93**
AU 25	0.79	**0.97**	**0.97**
AU 27	0.87	**1.00**	**1.00**
Average	0.575	0.926	**0.939**

There is no strict relationship between the frequency of AUs and their F1-score, and the classes that presented low F1-score (A15, A17, and A20) do not necessarily have a low frequency (89, 196, and 77 samples, respectively). All AUs with the lowest F1-score are from the lower face (lip corner depressor, chin raiser, and lip stretcher). Probably the error of such classes comes from the fact that they have very similar features, whose nuances and idiosyncrasies were not preserved by the preprocessing techniques used.

Although WiSARD had an overall F1-score lower than the other methods, it does the classification with a single frame, which in certain domains may actually be the only one available. In this way, we can observe how WiSARD is a broader solution, with online training and unrestricted use in this field, although it is not able to correctly classify some classes satisfactorily.In order for WiSARD to be able to perform well on these classes, it should be able to learn the intermediate representations of the data. We do not have information on the training and testing times for the state-of-the-art, so a complete comparison is not possible.

3.4.5 ClusWiSARD in Unsupervised Tasks

Experiments were also performed using the annotated images of the CK+ dataset with ClusWiSARD in unsupervised mode to validate its clustering power in this dataset, in an attempt to better interpret the use of ClusWiSARD in the semi-supervised mode in the experiments described here.

To verify if the problem of that approach was the ability of ClusWiSARD to select the best discriminator for non-annotated data, another test was done using a ClusWiSARD in unsupervised mode. Here, the net has one class and no restrictions on the number of discriminators, and each example was trained exclusively in one discriminator. In this mode, an example is learned by a discriminator if it is the one with the highest score in classification mode among all discriminators that satisfy ClusWiSARD's criterion for learning. In the case of ties between discriminators, the tie-breaking policy is to increase the bleaching and if this cancels the score of all discriminators candidates before the tiebreaker occurs, the larger discriminator is elected to learn the example. If no discriminator satisfies the learning criterion, a new discriminator is created to learn the example.

All the annotated images were used without their labels, which were then used to evaluate the clustering potential with Rand, Jaccard, and Folkes-Mellow metrics. $Rand = \frac{a+d}{a+b+c+d}$, $Jac = \frac{a}{a+b+c}$ and $FM = \frac{a}{\sqrt{(a+b)(a+c)}}$, where a are the number of pairs that belong to the same class and even cluster, b the quantity of those belonging to the same class and different clusters, c the number of those belonging to different classes and even cluster and d the quantity of those belonging to different classes and different clusters, and each multi-label example being decomposed into several single labels. The results using a ClusWiSARD with $n = 5$, $s = 0.1$ and $\varphi = 1000$ were Rand = 95.7%, Jac = 95.7% and FM = 97.8%, which indicate good performance of ClusWiSARD for unsupervised, and consequently, semi-supervised learning, thus leading to the conclusion that the non-annotated CK+ data did not have expressive enough AUs, having it probably been obtained in moments of transition of emotions, when AUs were far from their apex.

3.4.6 Discussion

The main difficulties encountered in the classification of AUs with WANNs were: (a) WiSARD and Label Powerset: few examples per discriminator; (b) ClusWiSARD and Label Powerset: most non-annotated examples were trained by discriminators who already had many examples; (c) WiSARD and Binary Relevance: many instances of an absolute tie between discriminators; depending on the adopted policy has low accuracy or low recall; high sensitivity to combinations of AUs; (d) all approaches: many false negatives.

3.5 Chapter Conclusion

This chapter presented novel approaches for classifying AUs activation with weightless neural networks. These results were published in [125]. A relevant contribution is the exploration of the WiSARD model in a hard multi-label problem, a yet non-observed feat in the literature, to the best of our knowledge. In 10-fold cross-validation, the best values found for accuracy and F1-score are, respectively, 89.66%

and 49.11%. WiSARD performed bad for Action Units whose the shape varies a lot between the samples. The speed of the proposed WANN architectures, both in training and classification phases, was of a very low order of magnitude, but a proper state-of-the-art comparison can not be provided because it use non-avaiable code, preventing proper reproduction of experiments.

Another contribution was a new way of using ClusWiSARD in unsupervised learning (others were presented in [127] and [126]), and the proof of its vocation for this type of task, even in a dataset where each example is highly idiosyncratic. Some related ongoing works are use of co-occurrence rules, ensemble between different approaches, separate upper and lower face AUs, other preprocessing techniques, non-annotated data filtering policy, and optimization of discriminator mapping.

Chapter 4

Ensemble Learning with WiSARD

Ensemble learning is a very valuable technique in machine learning due to its ability to combine several models in a single committee, in such a way that it tends to have greater accuracy in classification and regression tasks than each of its models individually, even that all of these models have a higher error rate in the ensemble learning. This occurs because the individual models learn fewer samples when training with only one subset[218].

One of the obvious disadvantages of using ensembles is the increase in training and classification time, which makes the use of weightless nets especially recommended for these committees. Despite their great vocation for online learning, WiSARD have been little explored in the literature in ensemble learning, and when they have been used, the structure of the committees is strongly oriented to the domain of the problem[177, 178, 219]. Unlike them, the ensemble types presented in this thesis can be used for any multi-class supervised learning task in a fully domain-agnostic way.

Although ClusWiSARD is traditionally used as an individual WANN, it can also be considered an ensemble of discriminators or a committee of networks that only have a single class pattern. Although ClusWiSARD's traditional classification policy involves using the score of each discriminator individually, without jointly observing the score of discriminators of the same class, in some domains other policies could be implemented in order to combine the outputs of these discriminators in order to reduce the standard deviation of intra-class classifications. This chapter presents new strategies for WiSARD ensembles and their exploration in several classification datasets.

4.1 Related Work

Ensemble learning are techniques used to solve classification, regression or clustering tasks that are based on generating a set of learning models and combining their results to obtain a more robust and accurate result than any of the models would obtain individually [220]. Ensembles can be effective in problematic machine learning issues, such as class imbalance, concept drift and curse of dimensionality[218, 221].

Ensembles distance themselves from divide-to-conquer strategies because they are more than simply dividing a dataset into smaller sets and applying different models to each of them. In ensemble learning each model is trained with a data subset with the possibility of subsampling or even with distinct features. Ensembles usually have a pruning step, where less important models on the committee are discarded.

Some fundamental types of ensembles are described in the next subsections. The comparison between is better visualized in Table 4.1.

4.1.1 Bagging

Bagging[222], or bootstrap aggregating, is an ensemble that generates several independent models trained from random redistribution of the original dataset. Each classifier in this ensemble is trained with the same number of examples in the training set, that is, models can learn the same example more than once.

It initially appeared as an attempt to reduce the variance of models that make the selection of variables and fitting in a linear model. Another original motivation for Bagging was to add robustness to machine learning algorithms where a small change in the training set produces great variation in predictions. Since models are independent, they can be trained in parallel.

A more formal mathematical description of a classification model ensemble would be:

Assume that we have pairs X_i, Y_i, where X_i corresponds to a single data in the dataset and Y_i corresponds to its respective class, where $Y_i \in \{0, 1, ..., C-1\}$ (classification with C classes). The target function is $P\left[Y = j|X = x\right](j = 0, 1, ..., C-1)$. The function estimator, which is the result from a given base procedure, is:

$$\hat{g}(\cdot) = h_n((X_1, Y_1), ..., (X_n, Y_n))(\cdot) : \mathbb{R}^d \to \mathbb{R} \tag{4.1}$$

where the function $h_n(\cdot)$ defines the estimator as a function of the data.

4.1.2 Arcing and AdaBoost

In the boosting ensemble[223], each classifier is built from the validation of the past classifiers, that is, after a model is trained it has its learning validated in the training set itself and the samples that were wrongly classified have more randomly selected in the resampling that will be the original training set of the next model. This

Table 4.1: A comparison between the most used ensembles in literature.

	Training stage	**Validation stage**	Inference stage
Bagging	Subset with replacement and repetition	-	Most voted
AdaBoost	Subset without replacement and repetition	Used for create new weak learners.	Most voted
Arcing	The probability of each example being drawn depends on previous validation.	Used for create new weak learners.	Most voted

validation can be used to generate weights for weak learners so that their votes contributed differently in the classification stage.

In particular, two models of Boosting are prominent in the literature, they have appeared previously: Arcing[224] and AdaBoost[225]. Similar to Bagging, Arcing draws with replacement N examples of a training set of size N for each classifier. Unlike Bagging, each example will not have the same probability of being drawn, but this probability will be dependent on how this example was classified by the models instantiated earlier. AdaBoost can select all the examples in the dataset and just assign a weight to them depending on their associated error.

Both models initialize the probability of choosing each example of the training set to be $\frac{1}{N}$ and then this probability is recalculated after a validation phase. In AdaBoost, each shift where a new classifier will be instantiated, the probability of integrating the training set of the examples wrongly classified by it is multiplied by the $\frac{(1-\alpha)}{\alpha}$ factor, where α is the sum of the current probabilities of these examples. After updating the probabilities, all of them are normalized so that their sum is equal to 1.

4.2 WiSARD Ensembles

All the ensembles proposed and listed below can be created from a combination of WiSARD, ClusWiSARDs, or both models, which will be called a heterogeneous ensemble. The address size of each neural network is determined randomly.

Bagging and Boosting based ensembles were chosen because the efficiency of such methods is proved in the literature. Since one of the goals here was not to sacrifice online training when using the ensemble, the validation stage of the traditional Boosting algorithm is not used to create new weak learners but rather to give weak learners rating weights. three other ensemble types were created in order to avoid the use of the bleaching algorithm on weak learners, in order to speed up the classification of individual models, specifically the WiSARD Borda Count is used in order to obtain the maximum likelihood estimator of the class ranking in a

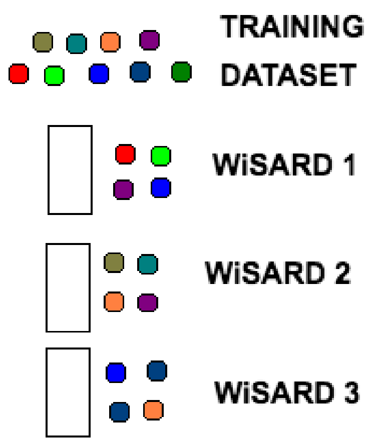


Figure 4.1: WiSARD Bagging's training datase

computationally cheap way.

4.2.1 WiSARD Bagging

In this ensemble, a set of weightless networks are used as weak learners and learn a subset of the training set and the response of the ensemble consists of the class with the most votes. In the event of a tie, one of the most voted classes is chosen arbitrarily in response to the ensemble. All weak learners are trained using exactly the same number of samples, which can be the same size as the original training set or just a partition of it, which are drawn with replacement. It is also possible to carry out the draw with repetition in the same subset so that the same sample can be trained more than once by the same network, as illustrated in Figure 4.1.

4.2.2 WiSARD Boosting

Ensembles where weak learners are trained with subsets obtained from the original training set without replacement and without repetition, as illustrated in Figure 4.2. 70% of the data of this subset are used for training and 30% for a validation phase, where the vote weight of each network is calculated based on a normalization of its accuracy.

In this ensemble, the choice of training subsets is not made based on validation, being validation used here only for calculating the weight of the vote of each weak learner in the ensemble's output. The purpose of this ensemble is to increase the degree of differentiation between the networks, making them specialists. The classification stage of this ensemble is exactly like that of WiSARD Bagging.

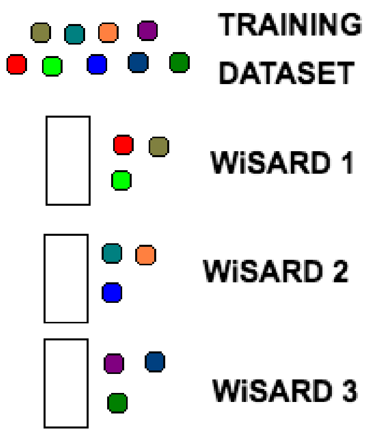


Figure 4.2: WiSARD Boosting's training datase

Table 4.2: Borda count starting at 1: scores received by each candidate from the ballot of a voter

Ranking	Candidate	Formula	Points
1st	T'Challa	p	5
2rd	Steve Rogers	p-*1*	4
3rd	Stephen Strange	p-*2*	3
4th	Peter Parker	p-*3*	2
5th	Scott Lang	p-*4*	1

4.2.3 WiSARD Borda Count

This ensemble is constituted and trained in the same way as WiSARD Bagging, however, it has a different classification system, based on diversified voting policies, based on the Borda count election system, formally proposed by the mathematician Jean-Charles de Borda[226]. In this ensemble, each weak learner will vote on all class options in the domain, giving them positions based on the scores they obtained in their classification phase, that is, the number of non-null memory locations that were accessed in each of its discriminators.

From here on we will use the nomenclature of voting systems for ensembles inspired by them, so the rank generated by each weak learner will be called *ballot*. In the event of a tie between the discriminators' scores, the sequence in the rank of the classes involved is chosen arbitrarily. These ballots will be used to calculate the ensemble's response according to any of the following policies:

4.2.3.1 Starting at 1

Each class will receive a score equal to the inverse of their position in the rank of each ballot. The class with the highest score in the entire ensemble is chosen as the system output. This classification system is showed in Table 4.2.

Table 4.3: Borda count starting at 0: scores received by each candidate from the ballot of a voter

Ranking	**Candidate**	**Formula**	**Points**
1st	Samwise	*p-1*	4
2rd	Bilbo	*p-2*	3
3rd	Meriadoc	*p-3*	2
4th	Peregrin	*p-4*	1
5th	Frodo	*p-5*	0

Table 4.4: Borda count in Dowdall system: scores received by each candidate from the ballot of a voter

Ranking	**Candidate**	**Formula**	**Points**
1st	Yoda	$\frac{1}{p-4}$	1.0
2rd	Obi-Wan	$\frac{1}{p-3}$	0.5
3rd	Chewbacca	$\frac{1}{p-2}$	0.33
4th	Han	$\frac{1}{p-1}$	0.25
5th	Luke	$\frac{1}{p}$	0.2

4.2.3.2 Starting at 0

Each class will receive a score equal to the number of classes below it in the ballot rank. The class with the highest score in the entire ensemble is chosen as the system output. This method penalizes the candidate in the last position on a ballot, as he will not receive any points from this voter. This classification system is showed in Table 4.3.

4.2.3.3 Dowdall

In this method, each class will receive $\frac{1}{p}$ points in each ballot, where p is its position in that rank. This method favors classes that have received many first preferences compared to the other two previous methods. This classification system is showed in Table 4.4.

4.2.4 WiSARD Tie-break Ensembles

These ensembles use the WiSARD Bagging structure for training and use classification policies that contemplate alternatives to deal with the tie, which was not the case in the others. In all of them, if there is a tie-up to the maximum number of possible turns, one of the classes involved in the last tie is chosen arbitrarily as the winner. Table 4.5 will be used to exemplify the classification policies used by this ensemble.

Table 4.5: This table is the result of a vote carried out by 20 weak learners, where the position of each class in the rank of each voter is given through the score obtained by him during the classification phase. In the event of a tie in the scores of the discriminators, the tiebreaker by individual rank occurs through arbitrary choice.

	1st	**2rd**	**3rd**	**4th**	**5th**
Leonard Hofstadter	5	4	4	2	5
Sheldon Cooper	3	6	0	4	7
Howard Wolowitz	5	4	4	4	3
Rajesh Koothrapalli	2	3	8	7	0
Bert Kibbler	5	3	4	3	5

4.2.4.1 All candidates

In this policy, in case of a tie in position i of the rank, a new round takes place using the votes of candidates in position $i + 1$ and all candidates are able to participate again.

In the case of Table 4.5, in the first round, we have a tie between "Leonard", "Howard" and "Bert", all with the highest score in the first position of the rank (5 points), so there is a need for a second round, using the votes received by the classes in the second rank class. This time, the winner is "Sheldon", with 6 points.

4.2.4.2 Only Ties

In this policy, in case of a tie in position i of the rank, a new round takes place using the votes of candidates in position $i + 1$ and only candidates involved in previous tie are able to participate again.

In the case of Table 4.5, in the first round, we have a tie between "Leonard", "Howard" and "Bert", all with the highest score in the first position of the rank (5 points), so there is a need for a second round, using the votes received by the classes in the second rank class. This time, candidates with the highest score are "Leonard" and "Howard" with 4 points. The result in third turn is equal to the second. In the fourth turn the winner is "Howard", with 4 points.

4.2.4.3 Tie-break with Threshold

In this policy, in case of a tie in position i of the rank, a new round takes place using the votes of candidates in position $i + 1$ and only candidates that scored more than a previously established threshold can participate in the next round. This threshold corresponds to a percentage of the total votes.

In the case of Table 4.5, assuming the threshold used is 15% of the points, in the first round, we have a tie between "Leonard", "Howard" and "Bert", all with the highest score in the first position of the rank (5 points), so there is a need for a second round, using the votes received by the classes in the second rank class. This round will be able to participate all who have had at least 15% of the votes, that is, at least three votes. Thus, "Leonard", "Sheldon", "Howard" and "Bert" were able to participate in the second round, and the winner was "Sheldon", with 6 points.

4.2.5 Weighted Votes Ensembles

This ensemble is built using the same structure and training method as the WiSARD Bagging, but in its classification, each class receives a score $v_i * (C - i + 1)$, where v is the number of votes in the i-th position and C is the total number of classes.

Using Table 4.5, we have the following score to the "Leonard" class: 5*5 + 4*4 + 4*3 + 2*2 + 5*1 = 62. Likewise, the results of the other classes are: "Sheldon": 54, "Howard": 64, "Rajesh": 60, "Bert": 60. The winner is "Howard".

4.2.6 Discussing the Tiebreaker Criteria for WiSARD Ensembles

Here I will discuss the tiebreaker criterion adopted by each of the ensembles introduced in this Section.

4.2.6.1 All candidates

"All candidates" use a counter-intuitive approach in allowing all candidates to move on to the next stages of the voting process, which has obvious disadvantages, such as ignoring the cumulative score of votes in the most privileged positions of the rank and privilege candidates who had few votes in the first rounds as draws occur.

This option was only considered because it reflects one of the configurations that bleaching has assumed historically and the purpose of this approach was only to validate within an ensemble a tiebreaker mechanism that has been used internally for a classifier previously.

4.2.6.2 Only-ties

Only-ties have the advantage of filtering candidates who received few votes in the first rounds, but there is a risk of discarding a good candidate if it receives slightly fewer votes than other candidates not so good in previous rounds, often for arbitrary reasons, such as a specific configuration of the random mapping of weak learners.

4.2.6.3 Tie-break with Threshold

Tie-break with threshold ensures that only candidates well placed in one round can participate in the tiebreaker in the next shift, but without applying as high a level of rigor as that of Only-ties, with the threshold being responsible for regulating the acceptance margin, thus mitigating the rejection of good candidates due to little difference in the score of accesses in previous rounds.

4.2.6.4 Weighted Votes Ensembles

Weighted Votes ensembles it has the advantages of not having to carry out several rounds of tiebreakers and generating a single score that takes into account all the votes at once, which is intuitively fairer.

4.3 Experimental Results

Following are the results of experiments carried out with WiSARD ensembles.

4.3.1 Datasets

Below is a presentation of the datasets used in the experiments reported in this Section. The choice of datasets was based on the following criteria: datasets commonly used in the literature, which explore different domains (color images, gray-scale images, text and tabular), allowing to see how different preprocessing impacts on the performance of the ensembles.

4.3.1.1 Cifar 10

The CIFAR-10 dataset[227] contains 60000 32x32 color images in 10 different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks). There are 6000 images of each class. The images found on this data are tiny and have rotation, different scales, and different backgrounds that can end up being considerable noise after the binarization process necessary to prepare the input for WiSARD.

4.3.1.2 CKP

The Extended Cohn-Kanade dataset[4] consists of 500 image sequences from 100 subjects, aged between 18 and 30 years. 65% of individuals are female. 82% of the individuals are Caucasian, 15% African American, and 3% Latino and Asian. All images are completely frontal and are all posed. The task solved here consists in classify the emotion in photos in the classes Neutral, Happy, Sad, Disgust, Fear, Angry, and Surprise. Another task involving the use of the WiSARD in CKP dataset is elaborated in Section 3.4.

4.3.1.3 MNIST

One of the most traditional datasets used to benchmark and validate machine learning models, MNIST[228] consists of a collection of digitized handwritten digits in 28x28 images (10 classes, 60000 examples in training set and 10000 in test set).

4.3.1.4 Fashion MNIST

Fashion MNIST[229] is an alternative version of MNIST that contains gray-scale 28x28 images of clothes. It has ten classes: T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot (60000 examples in the training set and 10000 in the test set).

4.3.1.5 IMDB

IMDB[230] is a dataset for binary sentiment classification with a set of 25000 highly polar movie reviews for training, and 25000 for testing. There is additional unlabeled data for use as well.

4.3.1.6 MovieLens

MovieLens data set[231] consists of 100000 ratings (1-5) from 943 users on 1682 movies, where each user has rated at least 20 movies. Datas provides simple demographic info for the users (age, gender, occupation, zip). The data is randomly ordered. Each entry consists of a tab-separated list of (user id | item id | rating | timestamp), where the timestamps are Unix seconds since 1/1/1970 UTC.

Follow the features of the base:

- data: The full dataset, 100000 ratings by 943 users on 1682 items;
- info: The number of users, items, and ratings in the dataset;
- item: Information about the items (movies); this is list of: movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once. The movie ids are the ones used in the dataset;
- genre: A list of the genres;

- user: Demographic information about the users; this is a tab-separated list of: user id | age | gender | occupation | zip code. The user ids are the ones used in the dataset.

- occupation: A list of the occupations.

4.3.1.7 State-of-the-art Models

Here I will indicate the state-of-the-art models for these datasets:

- **Cifar10:** EffNet-L2 - 99.70%[232];

- **CKP:** CNN - 99.60%[233];

- **MNIST:** Branching/Merging CNN + Homogeneous Vector Capsules - 99.87%[234];

- **Fashion MNIST:** Fine-Tuning DARTS - 96.91%[235];

- **IMDb:** NB-weighted-BON + Dv-cosine - 97.4%[236].

4.3.2 Experimental Setup

The experimental environment used here is an Intel Core i5 1.8 GHz with 8 GB DDR. The ReW and CReW implementations used here are available, along with other weightless models, in the C++/Python wisardpkg library[1]. The ensemble implementations are in Python. All datasets were previously preprocessed in other environments and their binarizations were stored in .wpkds objects, so the training and test times listed here do not include the preprocessing time.

All image datasets - Cifar10, CKP, and Fashion MNIST - were preprocessed using Local Threshold, Mean Threshold, Otsu, and Yen's Binarizations. Cifar10 and CKP were converted to gray-scale before the preprocess. MNIST was preprocessed using Mean Threshold and Deskewing. IMDB was preprocessed with tf-idf in previously selected features. MovieLens was binarized with the Mean Threshold. All experiments were run three times.

4.3.3 Experimental Results

About the datasets: Cifar10, Fashion MNIST, and MNIST are balanced datasets, with CIFAR10 presenting 5000 copies of each class in the training set and 1000 copies of each class in the test set. Fashion MNIST and MNIST have 6000 and 1000, respectively.

[1]Detailed in Appendix G

CKP has 7 classes and the number of copies in the training and test sets are, respectively: Angry - 36, 9; Disgust - 10, 8; Fear - 45, 14; Happy - 33, 3; Sadness - 58, 11; Surprise - 23, 6; Contempt - 67, 16. The great difficulty of this dataset lies in the small number of copies in the training set and the fact that many expressions of the same class are very heterogeneous.

IMDb and MovieLens are both binary classification datasets, where IMDb is balanced, with 12500 copies in each class (positive and negative), both in training and in the test set. And MovieLens has two classes, "F" and "M", with 217 copies in the training set, 56 in the test set, and 537 in the training set and 133 in the test set, respectively.

In these experiments, WiSARD and ClusWiSARD were tested with all address sizes in the range [5, 31], and for ClusWiSARD the *min score* and *growth interval* parameters were kept fixed at 0.1 and 100, respectively. The maximum number of discriminators per class of ClusWiSARD was varied in the range [3, 5]. All ensembles were tested with a composition of 10 and 20 learners and those based on the Bagging structure were tested with both 60% and 80% of the training set. All Borda Count and Tie-break ensembles policies were used. The comparison of accuracy, training time, and test time for WiSARD and ClusWiSARD, with its three possibilities of discriminator limits, is shown in Appendix C. The main results of each model in each dataset are shown in Tables 5.5-5.10 and the complete results of the experiments are found in Appendix B.

4.3.4 Discussion

Some considerations about the experiments described in 4.3.3.

- Mean threshold and Otsu's Binarization were the best preprocessing in all image datasets;

- In Cifar10 dataset the best models or ensembles in each binarization were: local threshold - WiSARD, mean threshold - Bagging and Borda Count (start at 0) ensembles, Otsu's Binarization - Bagging, Yen's Binarization - Borda Count (start at 1). The best model was Bagging;

- In CKP dataset the best results were: local threshold - WiSARD, mean threshold - Tie break (with threshold) and Weighted Votes ensembles, Otsu's Binarization - Borda Count (dowdall) and Tie-break (with threshold), Yen's Binarization - Bagging. The best model was Tie-break (with threshold);

- In Fashion MNIST dataset the best results were: local threshold - Bagging and Borda Count (start at 0) ensembles, mean threshold - Bagging and Borda

Table 4.6: The best results per model in Cifar10 dataset. Abbreviations: PP - preprocessing, n - tuple size, Pt - partition, training set size used, w; - number of weak learners, TrT - training time; LT - Local Threshold, MT - Mean Threshold, OB - Otsu's Binarization, YB - Yen's Binarization, WSD - WiSARD, Clus - ClusWiSARD, Bg - Bagging, BC - Borda Count, Tb - Tie-break, WV - Weighted Votes.

PP	**Mod**	**n**	**Pt**	**wl**	**Pol**	**Accuracy**	**TrT**	**Test time**
LT	**WSD**	14	-	-	-	**0.27 ± 0.00**	1.26 ± 0.00	1.96 ± 0.01
	Clus	14	-	-	M5	0.27 ± 0.02	2.46 ± 0.01	3.70 ± 0.06
	Bg	-	0.6	10	Mix	0.24 ± 0.02	10.75 ± 0.67	61.49 ± 11.07
	Boost	-	-	10	Mix	0.22 ± 0.02	48.89 ± 49.98	112.96 ± 30.64
	BC	-	0.6	20	Dwd	0.23 ± 0.04	13.87 ± 1.26	53.95 ± 10.37
	Tb	-	0.6	10	Th	0.18 ± 0.01	5.86 ± 0.29	22.44 ± 2.48
	WV	-	0.6	10	-	0.18 ± 0.04	6.10 ± 0.42	23.12 ± 0.38
MT	WSD	16	-	-	-	0.31 ± 0.00	**1.12 ± 0.01**	**1.84 ± 0.01**
	Clus	17	-	-	M3	0.31 ± 0.00	1.62 ± 0.02	2.73 ± 0.08
	Bg	-	0.6	20	Clus	**0.34 ± 0.01**	34.70 ± 4.41	137.29 ± 36.64
	Boost	-	-	10	Mix	0.28 ± 0.03	27.53 ± 15.79	99.035 ± 15.93
	BC	-	0.8	20	St0	**0.34 ± 0.01**	17.84 ± 0.59	73.54 ± 20.31
	Tb	-	0.6	20	Ot	0.24 ± 0.01	10.92 ± 0.69	54.05 ± 0.91
	WV	-	0.6	20	-	0.22 ± 0.02	10.58 ± 0.34	56.63 ± 6.18
OB	WSD	16	-	-	-	0.32 ± 0.00	**1.05 ± 0.00**	**1.83 ± 0.01**
	Clus	18	-	-	M3	0.31 ± 0.00	1.63 ± 0.02	2.86 ± 0.01
	Bg	-	0.8	20	WSD	**0.36 ± 0.01**	13.23 ± 0.77	46.52 ± 9.38
	Boost	-	-	10	Mix	0.30 ± 0.02	57.22 ± 62.84	110.43 ± 63.64
	BC	-	0.6	20	St1	0.35 ± 0.01	10.51 ± 0.49	73.71 ± 28.59
	Tb	-	0.6	20	All	0.23 ± 0.02	10.47 ± 0.68	50. ± 3.93
	WV	-	0.8	20	-	0.24 ± 0.00	13.75 ± 0.99	54.90 ± 5.66
YB	WSD	16	-	-	-	0.31 ± 0.00	**0.99 ± 0.00**	**1.79 ± 0.01**
	Clus	17	-	-	M3	0.30 ± 0.00	1.64 ± 0.03	3.33 ± 0.12
	Bg	-	0.8	20	WSD	**0.35 ± 0.00**	9.98 ± 0.60	41.66 ± 9.55
	Boost	-	-	10	WSD	0.30 ± 0.01	36.03 ± 22.70	122.49 ± 33.98
	BC	-	0.8	20	St1	**0.35 ± 0.00**	14.21 ± 0.69	74.45 ± 25.80
	Tb	-	0.8	20	Ot	0.23 ± 0.01	13.29 ± 0.85	48.00 ± 1.69
	WV	-	0.6	20	-	0.23 ± 0.01	10.19 ± 0.52	23.75 ± 2.80

Table 4.7: The best results per model in CKP dataset; Pol - Policy; M - Max; St1 - Start at 1; Ot - Only ties; Dowdall - Dwd; Threshold - Th.

PP	Mod	n	Pt	wl	Pol	Accuracy	TrT	Test time
LT	WSD	13	-	-	-	**0.51 ± 0.04**	**0.03 ± 0.00**	**1.96 ± 0.01**
	Clus	12	-	-	M5	0.49 ± 0.02	0.09 ± 0.00	3.70 ± 0.06
	Bg	-	0.8	20	WSD	0.45 ± 0.02	0.36 ± 0.03	0.54 ± 0.026
	Boost	-	-	10	Mix	0.33 ± 0.06	0.19 ± 0.02	2.21 ± 0.06
	BC	-	0.6	20	St0	0.42 ± 0.05	0.39 ± 0.01	53.95 ± 10.37
	Tb	-	0.8	20	Th	0.47 ± 0.03	0.54 ± 0.03	22.44 ± 2.48
	WV	-	0.8	20	-	0.47 ± 0.11	0.56 ± 0.06	23.12 ± 0.38
MT	WSD	13	-	-	-	0.57 ± 0.01	**0.02 ± 0.00**	**1.84 ± 0.01**
	Clus	14	-	-	M3	0.57 ± 0.02	0.06 ± 0.00	2.86 ± 0.01
	Bg	-	0.8	10	WSD	0.55 ± 0.03	0.16 ± 0.03	0.26 ± 0.01
	Boost	-	-	10	WSD	0.52 ± 0.04	0.20 ± 0.02	2.03 ± 0.03
	BC	-	0.8	10	St0	0.57 ± 0.03	0.26 ± 0.02	73.54 ± 20.31
	Tb	-	0.6	20	Th	**0.62 ± 0.09**	0.40 ± 0.01	54.05 ± 0.91
	WV	-	0.8	20	-	**0.62 ± 0.08**	0.49 ± 0.03	56.63 ± 6.18
OB	WSD	15	-	-	-	0.54 ± 0.02	**0.02 ± 0.00**	**1.83 ± 0.01**
	Clus	6	-	-	M3	0.55 ± 0.01	0.10 ± 0.00	2.73 ± 0.08
	Bg	-	0.6	20	WSD	0.54 ± 0.01	0.26 ± 0.01	0.48 ± 0.03
	Boost	-	-	10	WSD	0.46 ± 0.05	0.21 ± 0.03	2.06 ± 0.02
	BC	-	0.8	20	Dwd	**0.56 ± 0.02**	0.49 ± 0.02	73.71 ± 28.59
	Tb	-	0.6	20	Th	**0.56 ± 0.05**	0.37 ± 0.02	50.80 ± 3.93
	WV	-	0.8	20	-	0.55± 0.04	0.51 ± 0.04	54.90 ± 5.66
YB	WSD	5	-	-	-	0.50 ± 0.02	**0.03 ± 0.00**	**1.79 ± 0.01**
	Clus	5	-	-	M4	0.50 ± 0.02	0.12 ± 0.00	0.21 ± 0.00
	Bg	-	0.6	20	WSD	**0.54 ± 0.01**	0.26 ± 0.01	0.45 ± 0.022
	Boost	-	-	10	WSD	0.40 ± 0.05	0.19 ± 0.02	2.04 ± 0.05
	BC	-	0.6	10	St0	0.43 ± 0.02	0.20 ± 0.01	74.45 ± 25.80
	Tb	-	0.8	20	All	0.43 ± 0.03	0.48 ± 0.01	48.00 ± 1.69
	WV	-	0.8	20	-	0.44 ± 0.07	0.48 ± 0.01	50.72 ± 2.09

Table 4.8: The best results per model in Fashion MNIST dataset; Mod - Models; Pol - Policy; M - Max; St1 - Start at 1; Ot - Only ties; Dowdall - Dwd.

PP	**Mod**	**n**	**Pt**	**wl**	**Pol**	**Accuracy**	**TrT**	**Test time**
LT	WSD	24	-	-	-	0.80 ± 0.04	**0.42 ± 0.00**	**0.72 ± 0.00**
	Clus	24	-	-	M5	0.81 ±	1.95 ± 0.07	3.19 ± 0.02
	Bg	-	0.8	10	WSD	**0.83 ± 0.00**	3.09 ± 0.05	18.66 ± 7.93
	Boost	-	-	10	Clus	0.82 ± 0.00	19.22 ± 4.45	96.34 ± 29.47
	BC	-	0.8	10	St0	**0.83 ± 0.01**	4.37 ± 0.07	26.75 ± 3.52
	Tb	-	0.8	10	All	0.75 ± 0.01	3.89 ± 0.08	17.12 ± 1.04
	WV	-	0.6	20	-	0.77 ± 0.00	5.70 ± 0.10	33.44 ± 3.56
MT	WSD	28	-	-	-	0.81 ± 0.00	**0.30 ± 0.00**	**0.54 ± 0.00**
	Clus	28	-	-	M4	0.82 ± 0.00	1.25 ± 0.13	2.32 ± 0.02
	Bg	-	0.8	20	WSD	**0.84 ± 0.00**	4.93 ± 0.07	30.42 ± 2.26
	Boost	-	-	20	WSD	0.82 ± 0.00	42.33 ± 10.97	168.16 ± 63.23
	BC	-	0.8	10	Dwd	**0.84 ± 0.01**	3.60 ± 0.14	18.34 ± 1.60
	Tb	-	0.6	20	Ot	0.78 ± 0.00	5.04 ± 0.07	34.58 ± 4.23
	WV	-	0.6	20	-	0.78 ± 0.01	5.16 ± 0.15	34.66 ± 3.14
OB	WSD	27	-	-	-	0.80 ± 0.00	**0.33 ± 0.01**	**0.62 ± 0.00**
	Clus	28	-	-	M5	0.81 ± 0.00	1.68 ± 0.13	3.13 ± 0.07
	Bg	-	0.6	20	Clus	**0.84 ± 0.00**	23.22 ± 4.19	152.88 ± 51.18
	Boost	-	-	10	WSD	0.82 ± 0.00	19.91 ± 4.69	54.71 ± 5.05
	BC	-	0.6	10	St0	0.83 ± 0.00	2.59 ± 0.16	24.51 ± 1.36
	Tb	-	0.6	20	Ot	0.77 ± 0.00	5.06 ± 0.09	34.61 ± 0.29
	WV	-	0.8	20	-	0.77 ± 0.01	5.19 ± 0.07	35.79 ± 3.68
YB	WSD	23	-	-	-	0.75 ± 0.00	**0.45 ± 0.00**	**0.74 ± 0.00**
	Clus	23	-	-	M4	0.76 ± 0.01	1.63 ± 0.01	2.77 ± 0.07
	Bg	-	0.6	20	WSD	**0.80 ± 0.00**	4.08 ± 0.24	20.57 ± 4.99
	Boost	-	-	10	WSD	0.77 ± 0.00	20.06 ± 6.67	72.97 ± 15.91
	BC	-	0.8	10	Dwd	0.79 ± 0.01	3.86 ± 0.06	22.40 ± 3.13
	Tb	-	0.6	20	Ot	0.69 ± 0.01	5.61 ± 0.13	31.36 ± 0.61
	WV	-	0.8	20	-	0.70 ± 0.01	7.63 ± 0.06	34.02 ± 4.81

Table 4.9: The best results per model in MNIST dataset.

Models	**n**	**Pt**	**wl**	**Policy**	**Accuracy**	**TrT**	**Test time**
WSD	30	-	-	-	89 ± 0.00	**0.61 ± 0.13**	**0.63 ± 0.00**
Clus	28	-	-	Max = 3	89 ± 0.00	0.88 ± 0.01	2.24 ± 0.034
Bg	-	0.6	10	WiSARD	**0.93 ± 0.00**	1.33 ± 0.05	10.36 ± 3.85
Boost	-	-	10	WiSARD	0.91 ± 0.00	20.50 ± 5.77	57.69 ± 13.04
BC	-	0.6	20	Start at 1	**0.93 ± 0.00**	4.50 ± 0.09	37.61 ± 11.32
Tb	-	0.8	20	Only	0.76 ± 0.02	6.15 ± 0.07	30.03 ± 2.03
WV	-	0.6	20	-	0.81 ± 0.01	4.76 ± 0.06	28.19 ± 3.15

Table 4.10: The best results per model in IMDb dataset.

Models	**n**	**Pt**	**wl**	**Policy**	**Accuracy**	**TrT**	**Test time**
WSD	5	-	-	-	0.59 ± 0.00	**1.49 ± 0.44**	**7.33 ± 0.28**
Clus	5	-	-	Max = 4	0.59 ± 0.00	4.94 ± 4.94	41.94 ± 5.68
Bg	-	0.6	20	Mix	0.70 ± 0.00	51.21 ± 5.78	22.39 ± 0.82
Boost	-	-	20	WiSARD	**0.77 ± 0.00**	21.08 ± 2.52	1329.95 ± 87.27
BC	-	0.6	20	Dowdall	0.70 ± 0.01	17.71 ± 1.92	190.16 ± 33.60
Tb	-	0.6	20	All	0.68 ± 0.05	18.22 ± 1.41	176.84 ± 21.13
WV	-	0.8	20	-	0.68 ± 0.03	24.11 ± 1.02	167.74 ± 29.76

Table 4.11: The best results per model in MovieLens dataset.

Models	**n**	**Pt**	**wl**	**Policy**	**Accuracy**	**TrT**	**Test time**
WSD	31	-	-	-	0.70 ± 0.02	**0.03 ± 0.00**	**0.01 ± 0.00**
Clus	30	-	-	Max = 5	**0.72 ± 0.01**	0.12 ± 0.00	0.06 ± 0.00
Bg	-	0.6	10	WiSARD	0.71 ± 0.00	0.17 ± 0.00	0.16 ± 0.02
Boost	-	-	10	WiSARD	0.70 ± 0.00	0.27 ± 0.05	4.67 ± 0.06
BC	-	0.6	10	Dowdall	0.70 ± 0.01	0.31 ± 0.02	0.72 ± 0.02
Tb	-	0.8	10	Threshold	0.46 ± 0.07	0.43 ± 0.03	0.70 ± 0.01
WV	-	0.6	10	-	0.45 ± 0.02	0.32 ± 0.02	0.69 ± 0.021

Count (dowdall), Otsu's Binarization - Bagging, Yen's Binarization - Bagging. The best model were Bagging and Borda Count (dowdall);

- In MNIST dataset the best models were Bagging and Borda Count (start at 1);
- IMDb dataset the best result was Boosting;
- In MovieLens dataset the best model was ClusWiSARD;
- In IMDb dataset all ensembles were more accurate than individual models;
- Considering only the ensembles, the best results in terms of training time were: Cifar10 - Tie-break and Weighted Votes, CKP - Boosting, Fashion MNIST - Borda Count, MNIST - Bagging, IMDb - Borda Count, MovieLens - Bagging;
- Regarding the test time: Cifar10 - Weighted Votes, CKP - Bagging, FASHION MNIST - Borda Count, MNIST - Bagging, IMDb - Bagging, MovieLens - Bagging;
- This comparison of training and testing time is not entirely appropriate, since the models that make up the ensemble are random, an ensemble may have had more ClusWiSARDs and therefore slower training. Not only structure of the ensemble and its policies influence time, but also the models that composed it and the size of their address;

- In general, it is expected that Bagging-based ensembles will have the fastest training, as it does not have the cost of validation, and that Borda Count, Tie-break and Weighted Votes will have the fastest classification since their weak learners do not perform bleaching, using only the scores of the discriminators of each weak learner and making the tiebreaker externally through the policies of the ensembles;

- In general, all ensembles had low standard deviation and variance both in accuracy, training and testing times. The exception is due to the IMDb dataset, where all models had a high standard deviation in all these metrics;

- No ensemble used here has been strictly optimized. While the models were programmed in C, all ensembles were done in Python. It would also be possible to perform in parallel the training and classification of all training models, including WiSARD Boosting, as it does not build each weak learner based on the performance of the previous one, different from traditional Boosting. It is also possible to perform in parallel the training of the different examples of the training set, since each training at WiSARD and ClusWiSARD is independent from the others;

- In Cifar10 dataset, the best ensemble outperforms WiSARD in 4% and ClusWiSARD in 5%;

- In CKP dataset, the best ensemble outperforms WiSARD and ClusWiSARD in 5%;

- In Fashion MNIST dataset, the best ensemble outperforms WiSARD in 3% and ClusWiSARD in 2%;

- In MNIST dataset, the best ensemble outperforms WiSARD and ClusWiSARD in 4%;

- In IMDb dataset, the best ensemble outperforms WiSARD and ClusWiSARD in 18%;

- In MovieLens, the best ensemble outperforms WiSARD in 1% and is outperformed by ClusWiSARD in 1%.

4.3.5 Additional Results in Cifar 10

In this subsection I will present additional results in Cifar 10. All models and ensembles used previously were used again. The difference here was in relation to

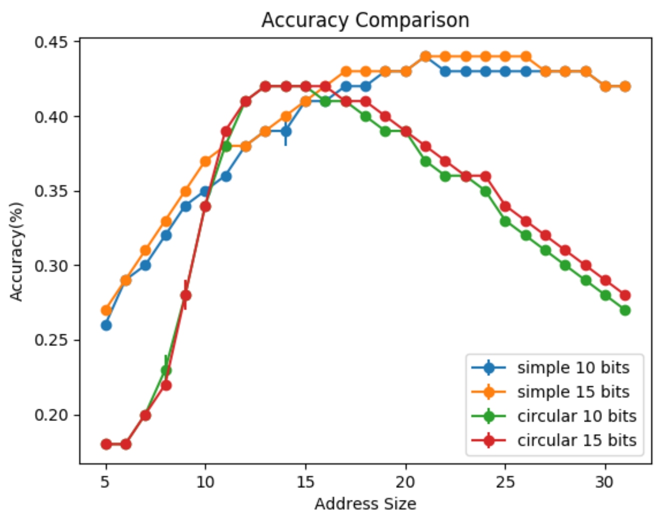


Figure 4.3: Comparison of WiSARD's accuracy in Cifar10 dataset with 10 and 15 bits in traditional and circular thermometer in RGB channels.

Table 4.12: The best results per model in Cifar10 dataset with traditional thermometer in RGB channels; Mod - Models; Pol - Policy;M3 - Max = 3; St1 - Start at 1; Ot - Only ties.

Mod	**n**	**Tl**	**Pt**	**wl**	**Pol**	**Accuracy**	**TrT**	**Test Time**
WSD	22	15	-	-	-	**43.66 ± 0.1**	37.00 ± 0.03	62.34 ± 0.02
Clus	17	15	-	-	M3	31.22 ± 0.11	271.69 ± 2.71	503.01 ± 12.02
Bg	-	10	0.8	20	WSD	42.34 ± 0.35	516.24 ± 22.21	1325.89 ± 102.68
Boost	-	10	-	10	WSD	39.25 ± 0.09	70.55 ± 2.96	2560.81 ± 101.04
BC	-	10	0.8	20	St1	42.98 ± 0.4	818.89 ± 41.77	1571.29 ± 57.23
Tb	-	10	0.6	20	Ot	32.49 ± 0.80	637.87 ± 15.74	1487.55 ± 43.33
WV	-	10	0.8	20	-	32.60 ± 0.50	606.85 ± 28.28	1431.69 ± 63.52

preprocessing: a thermometer is applied to each color channel (R, G, B) and all data words are concatenated to form the network input.

A variation of the traditional thermometer was used here. This is the circular thermometer, which has half its content filled with "0" s and the other half with "1" s, both of which are contiguous. The minimum value of the thermometer and any value less than that that eventually appears in the test set will be represented with the first "0" occupying the first bit of the thermometer. As the value of the number to be converted exceeds the threshold $\frac{maxValue - minValue}{thermometerSize}$ the representation of the thermometer will be shifted to the right. The motivation for this variation is an attempt to reduce the sparsity of neural network.

Both the traditional and circular thermometers were tested, using 10 and 15 bits per channel. The results of the WiSARD are shown in the Figures 4.3-4.5 and the benchmark of all models and ensembles is shown in the Tables 5.11 and 5.12. Three substantial differences from previous experiments: (i) the training and testing time

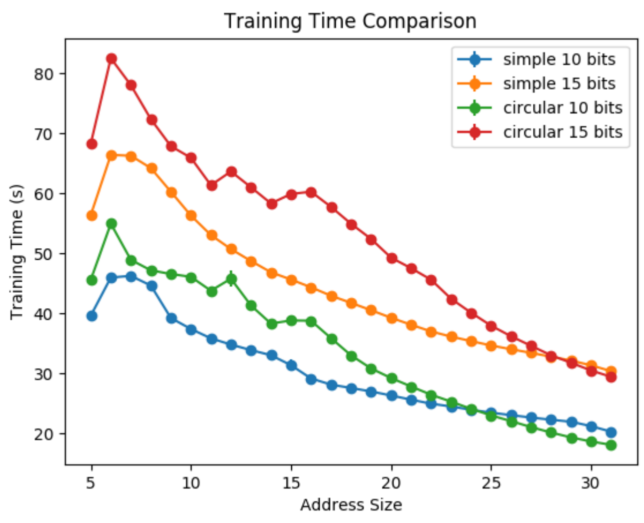


Figure 4.4: Comparison of WiSARD's training time in Cifar10 dataset with 10 and 15 bits in traditional and circular thermometer in RGB channels.

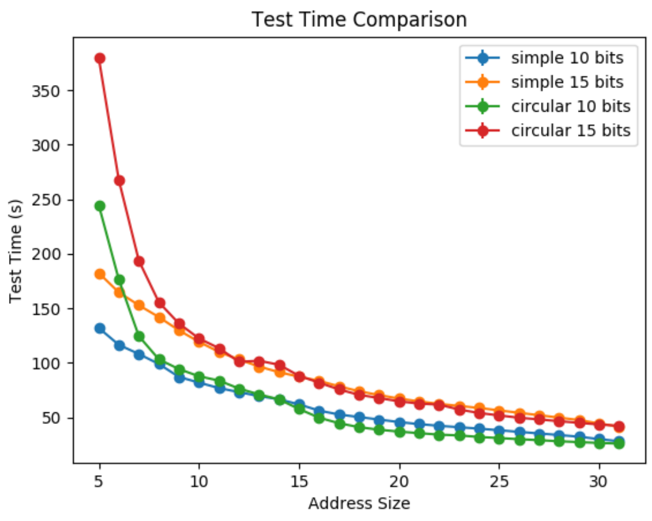


Figure 4.5: Comparison of WiSARD's test time in Cifar10 dataset with 10 and 15 bits in traditional and circular thermometer in RGB channels.

Table 4.13: The best results per model in Cifar10 dataset with circular thermometer in RGB channels; Mod - Models; Pol - Policy;M3 - Max = 3; St1 - Start at 1; Ot - Only ties.

Mod	**n**	**Tl**	**Pt**	**wl**	**Pol**	**Accuracy**	**TrT**	**Test Time**
WSD	22	15	-	-	-	**42.19 ± 0.2**	58.33 ± 0.24	98.26 ± 1.12
Clus	17	10	-	-	M3	19.38 ± 0.10	136.15 ± 6.02	241.92 ± 9.26
Bg	-	10	0.8	20	WSD	41.67 ± 0.00	502.00 ± 0.00	1390.00 ± 0.00
Boost	-	10	-	10	WSD	37.33 ± 0.70	63.63 ± 9.20	2062.79 ± 146.83
BC	-	10	0.8	20	St1	42.29 ± 0.00	931.68 ± 0.00	1869.60 ± 0.00
Tb	-	10	0.6	20	Ot	31.86 ± 3.70	714.42 ± 26.16	1799.27 ± 353.78
WV	-	10	0.8	20	-	26.06 ± 2.06	927.16 ± 66.10	2729.95 ± 1698.17

has increased; (ii) accuracy has increased (7% increase); and (iii) WiSARD won out of all ensembles. The increase in accuracy was probably due to the use of more features from each pixel. This can be useful to the limitation of WiSARD in dealing with variances related to rotation, translation, and scale. This can be a valid alternative as long as a more robust multi-layer solution that learns variations of the features and their correlations is not implemented or while another network topology is not structured is add to model to able this to deal with variance. One possibility consists in use a NC-WiSARD-like model[158] with convolution layers.

4.4 Chapter Conclusion

Based on traditional Bagging and Boosting ensemble learning techniques, two types of ensembles using WiSARD and ClusWiSARD were created. Based on the structure of WiSARD Bagging, three other types of ensembles were created. Its weak learners do not perform tie-break internally, only generating ballots from the scores of their discriminators. These ensembles are based on traditional voting systems. All these types of ensembles were tested in six datasets and compared with the performance of the models that compose them, evaluated individually. In some types of preprocessing WiSARD and ClusWiSARD performed better than ensembles, probably because no pruning technique was used, so that a combination of models can end up propagating error. However, in most cases, some type of ensemble performed better. Especially in the IMDb dataset, which was preprocessed using *tf-idf*, all ensembles had better accuracy than the individual models and the winning ensemble, WiSARD Boosting, had 18% more accuracy than the best WiSARD and ClusWiSARD configuration, using only 20 WiSARDs. It is worth mentioning that no technique was applied in the selection of the models used and in their parameters.

Ongoing works: adding pruning policies to the ensembles to remove networks that have poor individual performance, combining different preprocesses in the same weak learner, using the traditional Boosting policy to create new weak learners, paralleling the training and classification process on weak learners, extend WiSARD Bagging and WiSARD Boosting to perform clustering, use WiSARD in ensembles that have other paradigms besides WANNs.

However, the main future work is to test the ensembles with statistical significance tests, such as McNemar's test or 5×2 cross-validation with a modified paired Student t-test, to ensure that the difference in skill scores is statistically significant, increasing the confidence in the interpretation of results. Since ensembles are much more computationally costly, the improvement in accuracy or other performance metrics must be considerably large.

Chapter 5

Extending WiSARD for Regression

Regression is a traditional and important machine learning task since there is a wide range of practical situations in the real world where it is necessary to predict values in a continuous space. Weightless Artificial Neural Networks (WANNs), due to their simple, RAM-based architecture, seem to be a suitable computational intelligence model for this type of task. Since WiSARD and its extensions are just classifiers and many important domains in which ML is applied are related to regression tasks, extending these models so that they can make predictions of continuous values is essential for them to achieve more relevance. This chapter describes the new WiSARD-based regressors and their advantages in relation to the classic weightless regressor. Both of them are useful for any type of task that can be approximated as a non-parametric regression problem.

In a precision agriculture scenario, it would be desirable that simple devices, such as small sensors, could perform regression. This chapter explores the use of WANNs in the KDD18 competition[237], a challenge which goal is to predict the palm oil harvest productivity of a set of 28 different production fields using data provided by an agribusiness company. The dataset contains information about palm tree varieties, harvest dates, atmospheric data during the development of the trees, and soil characteristics of the fields where the trees are located in. The WANN models explored in this work are based on the n-tuple Regression Network[134], which was proved to be successful when compared to other classical regression approaches in non-linear plant approximation[145] and Mackey-Glass chaotic time series prediction tasks[238]. Here, a wider theoretical background is presented, alongside a broader exploration of their parameters and how the models perform when combined as ensembles.

The content of this text is organized as follows. Section 5.2 presents the two weightless models proposed for regression, and the ensemble techniques explored. Section 5.3 discusses the various approaches used in the KDD18 competition, as well as a comparison with state-of-the-art methods. This section also contains the description of experiments using the new models in the House Prices, CalCOFI, and Parkinson datasets. Concluding remarks and ongoing work are presented in Section 5.4.

5.1 Related Work

The most used non-parametric regression techniques in machine learning are:

- ***k*-NN[239]:** a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel. This model stores the entire training dataset and compute the Euclidean distance from the sample to be predicted to the training samples. Then, it orders the training samples by increasing distance and find a heuristically optimal number k of nearest neighbors, based on RMSE in cross validation. Then, calculate an inverse distance weighted average with the k-nearest multivariate neighbors;

- **XGBoost[240]:** a scalable regression tree boosting for multiple scenarios. This model can scale to billions of examples in distributed or memory-limited settings. XGBoost has become popular as the state-of-the-art solution to many of Kaggle's challenges in recent years. It is very useful for tabular data, however it is not very effective with unstructured data;

- **Support Vector Regression[241]:** a SVM-based model that depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

The use of n-tuple classifier-based models for non-parametric regression was explored in literature[134, 145, 146]. It is more elaborated in Section 2.5. The main limitation of this model is to use only a single kernel estimator to approximate *pdf* in any domain. This comparison is better visualized in Table 5.1.

5.2 The New Weightless Regression Models

5.2.1 Regression WiSARD

Regression WiSARD (ReW) is an extension of the n-tuple Regression Network, which adds to its original structure some characteristics of the WiSARD. There is a description of its general architecture below.

- Each RAM location in the ReW model has two dimensions: a counter and a $sum(y)$, a value formed by sum of the predictions learned by the network, both updated at each pattern training; initially all values are set to zero;

Table 5.1: Comparison between the main regressors in literature and non-parametric weightless regressor.

	Regression	**Kernel estimator**
***k*-NN**[239]	Stores the entire training dataset and compute the Euclidean distance from the sample to be predicted to the training samples.	Kernel density "balloon" estimator with a uniform kernel.
XGBoost[240]	Training Loss + Regularization	-
SVR[241]	Uses a curve as a linear boundary.	Kernel trick
GRNN[144]	Stores the entire training dataset and compute kernel estimator from the sample to be predicted to the training samples.	Nadaraya-Watson estimator
***n*-tuple Regression Network**[134]	Stores in a implicit way the entire training dataset and compute kernel estimator from the sample to be predicted to the training samples.	Nadaraya-Watson estimator

- ReW accepts binary data with exactly the size of its retina ($N * n$) as input, which normally requires some kind of preprocessing to transform the input data into a binary representation; each pseudo-randomly mapped group of n bits of the input retina will access the position corresponding to its values a neuron (RAM node);

- When during the prediction phase, a memory location that was never accessed during the training phase is accessed, ReW will respond as a "don't know answer" prediction, with the architecture of each system where ReW is used to handle this response according to the domain of the problem. In this work, a prediction 0 is made in cases of "don't know answer";

- ReW uses WiSARD's minZero and minOne.

5.2.1.1 Training

In the training phase (Fig. 5.1), k pairs $(\mathbf{x}_i, y_i)$ are submitted to the ReW network, and each of their corresponding addressed memory locations will have their two values updated; the counter is incremented and partial access is summed with the y_i of the example that generated the access.

5.2.1.2 Prediction

In the prediction phase the sum of counters ($\sum c$) and partial y ($\sum y$) of the positions accessed by a given $\mathbf{x}$ are used to calculate the corresponding y (Fig. 5.2); unlike the

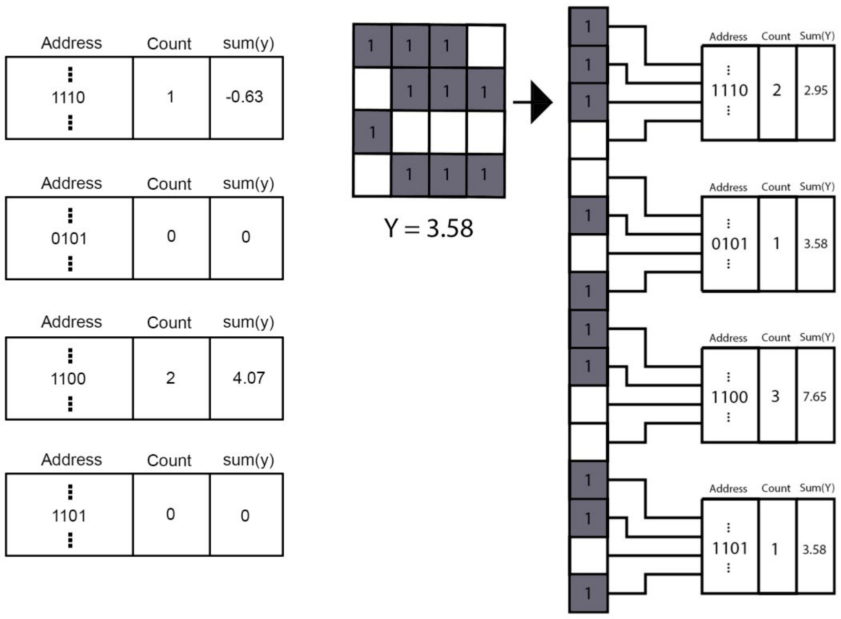


Figure 5.1: Example of a ReW model behavior in the training phase. A binary input and a float value y are presented to the model. The pseudo-random mapping is applied to the binary input and the new pattern is divided into n-tuples, each one being assigned to one of the regression RAMs. The values related to the address corresponding to the tuple are updated in the following way: the counter is incremented by 1, while the summation is incremented by the value of y.

n-tuple Regression Network that uses only simple mean ($\frac{\sum y}{\sum c}$) for this calculation, ReW can also use:

- power mean: $\sqrt[p]{\frac{\sum_{i=0}^{n}(\frac{y_i}{c_i})^p}{n}}$
- median: central value of $\frac{y_i}{c_i}$, with i in range [0, n])
- harmonic mean: $(\frac{\sum_{k=1}^{n}\frac{y_i}{c_i}^{-1}}{n})^{-1}$
- harmonic power mean: $\sqrt[p]{\frac{n}{\sum_{i=0}^{n}\frac{1}{(\frac{y_i}{c_i})^p}}}$
- geometric mean: $(\prod_{i=0}^{n}\frac{y_i}{c_i})^{\frac{1}{n}}$
- exponential mean: $log(\frac{\sum_{i=0}^{n}e^{\frac{y_i}{c_i}}}{n})$

Associated with each type of mean is an influence on the response set of RAMs. The median allows escape from the influence of outliers and the other mean types favor the influence of the contribution of memory locations that were most accessed during training, with different degrees of intensity (Harmonic, Power, Harmonic Power, and Geometric Mean, in ascending order). The Arithmetic Mean was kept since it was adopted by the original n-tuple Regression Network and is the only one that does not differentiate among RAM responses.

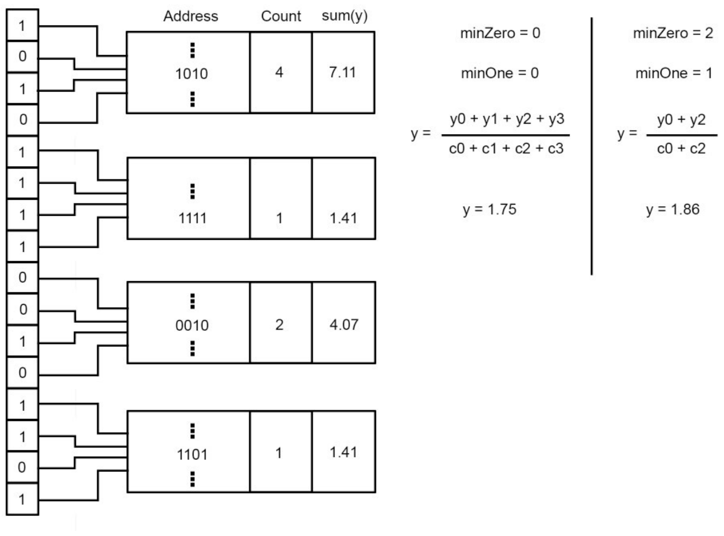


Figure 5.2: Prediction of the same example with different minZero and minOne values (simple mean).

Arithmetic Mean approximates the Nadaraya-Watson kernel regression estimator, as seen in Section 2.5.1. All of these means are based on the sum of counters and partial ys and therefore share the same kernel used in the original n-tuple regression method. Other means alter the kernel estimator, which together with the smoothing factor (amount of RAMs) changes how the neural network estimates *pdf*. However, to change the kernel, allowing the network to have greater opportunities to cover a wider range of possible functions, it would be necessary to change the way of using the contents of the memory locations, which is not addressed in this work.

This possibility to change the kernel estimator is the biggest difference and also the biggest contribution of Regression WiSARD in relation to the n-tuple Regression Network, since topologically and in terms of training algorithm, both models are identical. Other minor differences include the possibility of ignoring memory locations formed by tuples composed only of zeros or the minZero and MinOne parameters.

5.2.2 ClusRegression WiSARD

Inspired by ClusWiSARD, the ClusRegression WiSARD (CReW) is a network formed by several ReWs, each with distinct mappings, but with retinas of the same size and same address size as well.

5.2.2.1 Training

In the training phase, when a pair $(\mathbf{x}_i, y_i)$ is submitted to the network, $\mathbf{x}$ is presented for each ReW, which behaves like a class discriminator of the WiSARD in the

Algorithm 1 ClusRegression WiSARD algorithm

```
1: procedure Training
2:     Require $r0$ = minimum score
3:     Require $\gamma$ = threshold growth interval
4:     Require $\mu$ = maximum discriminators
5:     Require $T$ = training data
6:     Ensure $CReW$ is a trained ClusRegression WiSARD regressor
7:     for each observation $o \in T$ do
8:         for each discriminator $d$ currently in $CReW$ do
9:
10:            if $score(d, o) \geq \min\left(N, r_0 + \frac{N|d|}{\gamma}\right)$ then
11:                ReW discriminator $d$ learns $o$
12:        if no ReW discriminator learned the observation $o$ and $size(CReW) < \mu$ then
13:            A new discriminator $d0$ is created
14:            $d0$ is added to the collection of ReW discriminators of CReW
15:            Discriminator $d0$ learns observation $o$
16:    Return CReW
```

classification phase. Each ReW will return a score obtained from the number of positions of its memories that have been accessed and have countered with a value greater than zero. All ReW discriminators that satisfy the learning policy are trained with ($\mathbf{x}_i$, y_i).

If an observation is submitted to CReW but does not meet the requirement to be learned by any of its discriminators and the threshold for creating new discriminators (if it has been established) has already been reached, then this observation will not be learned because it is likely to be an outlier. The CReW training algorithm is detailed in Algorithm 1.

5.2.2.2 Prediction

In the prediction phase, when an input $\mathbf{x}$ is submitted to the CReW, it will be sorted by each ReW and the highest score will predict its corresponding y. If there is a tie between the ReWs, the tie-break policy known as bleaching, native to WiSARD, will be used. In it, a threshold initialized with value zero is incremented with each tie and a new classification occurs, being considered for the score of each discriminator only the memory locations whose counter is superior to the bleaching. Just like WiSARD, if there is an absolute tie, that is, the value of the bleaching is greater than the cardinality of the training set used, a previously chosen ReW default is elected.

5.2.3 Regression WiSARD Ensembles

To improve the predictive power of the new models, ensembles formed exclusively with ReW and CReW were also tested.

Three ensemble models were tested using ReW and CReW: Naïve (all models are trained with the entire training dataset and there is no restriction on the existence of fully redundant models), Bagging and Boosting. The weak learners are obtained by simple draw, and in the case of ReW the following parameters are drawn: address size (in the range [5, 32]), type of mean, minZero and minOne. For CReW, in addition to the parameters used in ReW are also randomized the minimum response, growth interval, and the maximum of ReW discriminators (in range [2, 6]).

In ReW Bagging, each weak learner is trained with subsets of the training dataset of the same size, with resampling. At training time, two parameters are selected: the number of weak learners and the size of the subset. ReW Boosting training also uses, for each weak learner, subsets of the same size, without resampling. 90% of the training subset is used for training and 10% for validation. The weight of the vote of each learner is determined by the normalization of its score in the validation phase.

Simple Mean, Median, and Harmonic Mean can be chosen to calculate the average of the individual predictions, resulting in ensemble prediction. These ensembles do not yet have any refined type of pruning, and ReW Bagging only discards strictly redundant learners, that is, learners with the same parameters trained with the same subset, and ReW Boosting and Naïve ReW Ensemble never discards any learner.

CReW can be considered an ensemble too. Ensembles don't necessarily need to combine individual predictions for generating a more accurate prediction. It can generate a prediction by choosing the best weak learner as CReW does.

5.3 Experimental Results

This section presents the results of ReW, CReW, and their ensembles in the dataset that motivated their creation: KDD18 dataset. Additionally, three other datasets were used in their validation. The experimental environment used here is an Intel Core i5 1.8 GHz with 8 GB DDR. The ReW and CReW implementations used here are available, along with other weightless models, in the C++/Python wisardpkg library[1]. All experiments were run three times.

[1]https://github.com/IAZero/wisardpkg

5.3.1 KDD18 Experimental Setup

The data available to the competitors was divided into three types of files: first, the training and testing files, containing 5243 and 4110 observations, respectively. Both files contain as features i) the id of the observation; ii) the id of the field the observation was planted; iii) the age of the palm tree [3, 26]; iv) the type of the palm tree; v) the year of harvest [2004, 2011]; and vi) the month of harvest [1, 12]. The training file also has information regarding the target y, which is the total amount of palm oil produced by the tree. Second, a file containing information regarding the soil properties of the field in which the palm tree is planted. Finally, 28 files containing historical data regarding weather measured in each field from January 2002 to December 2007. The production field, i.e, the y to be predicted, varies between 0 and 1.

The initial modeling removes the *id* and the $field\_id$ and adds additional information from the other files. First, a time window is defined to search weather information in a specific period going backward from the month before the harvest of the tree. Second, all 66 features related to the soil data are added. The new observation is then composed of 68 features (age, type, and 66 ground-related features) plus 8 features for each month contemplated by the time window.

In the second round of experiments, variations of the initial modeling were performed. One of them ignores the soil data by creating a total of 28 ReWs, each one responsible for predicting the production of trees planted in a specific field. Other variations aim to overcome the problem of the *type* feature: there are values in the testing file that are not present in the training file. These variations included the removal of the feature and the usage of one-hot encoding of all possible values.

Since the features must be binarized, a thermometer encoding is applied. Due to the short space of time, it was not possible to perform experiments aiming for the best thermometer value for each feature. As a result, the same value was applied to all features. However, a small set of different values were used for an empirical evaluation. In addition, since a binary word of size w can be divided into different sizes of n tuples, all possible n values that are less than 32 were tested.

5.3.2 Analysis of the KDD18 Experiments

The best of RAM-based solutions reached the seventh position of 51 teams[2].

Since the KDD18 test set annotations are not public, it was necessary to submit the results of the experiments to Kaggle to obtain their respective MAE and how the Kaggle API behaves problematically, losing results and even disrupting when a large flow of results is submitted, the experiments for KDD18 were not performed

[2]https://www.kaggle.com/c/kddbr-2018/leaderboard

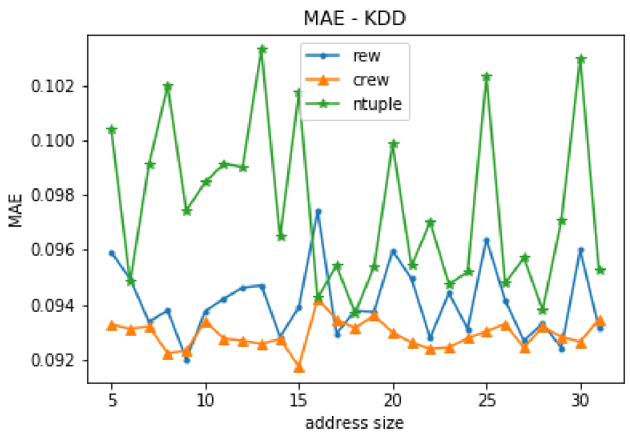


Figure 5.3: Address size X MAE for ReW, CReW and n-tuple Regression Network in KDD18 dataset.

multiple times to obtain the standard deviation (except for ensembled tests, where each had 10 rounds). Since the data is private, it was also not possible to measure the result with any other metric than that used in the challenge, MAE.

A dataset exploration varying the address size for the ReW, CReW and n-tuple Regression Network models and the amount of ReW Bagging, ReW Boosting and Naïve ReW Ensemble learners with their respective MAE, training time and test time obtained can be found in the Figs. 5.3, 5.4, 5.5, 5.6.

The best results obtained by the weightless regression models and their ensembles are compared with the state-of-the-art of this task and other relevant results in the Kaggle challenge in Table 5.2.

Table 5.2: Comparison of WANN regressor models with state-of-the-art in Private Score of KDD18 Challenge

Model	MAE	Training time (s)	Test time (s)
XGBoost	**0.07983**	4.12962484	0.08239889145
GradientBoost	0.08239	3864.08913588	0.00241994858
n-tuple Regression	0.09211	0.0037262439727	0.000348329544
RegressionWiSARD	0.09097	**0.00035619736**	**0.00017619133**
ClusRegressionWiSARD	0.09173	0.00040984154	0.00021290781
Naïve ReW Ensemble	0.08814	71.24	3523.83
ReW Bagging	0.13867	58.4	3308.82
ReW Boosting	0.14996	5.16	4689.94

In the Public Score of the KDD18 competition [237], all results were obtained from the validation dataset. ReW and CReW obtained MAE of 0.08737 and 0.08938, respectively, while XGBoost[240], the state-of-the-art, obtained MAE of 0.07569. A Naïve ReW Ensemble got MAE of 0.08468. The following experiments will only take into account results obtained with the test dataset, the Private Score of the

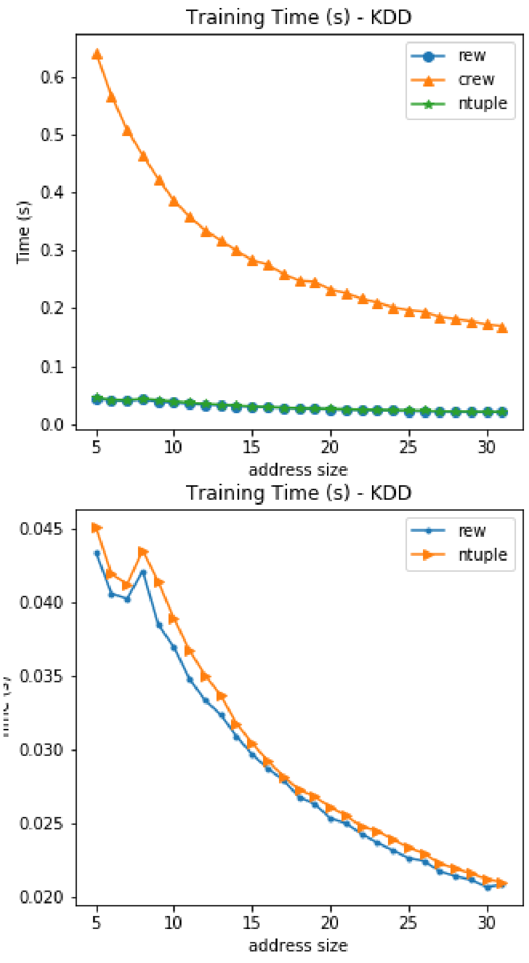


Figure 5.4: Address size X training time (s) for: (a) ReW, CReW and n-tuple Regression Network in KDD18 dataset; (b) ReW and n-tuple Regression Network in KDD18 dataset.

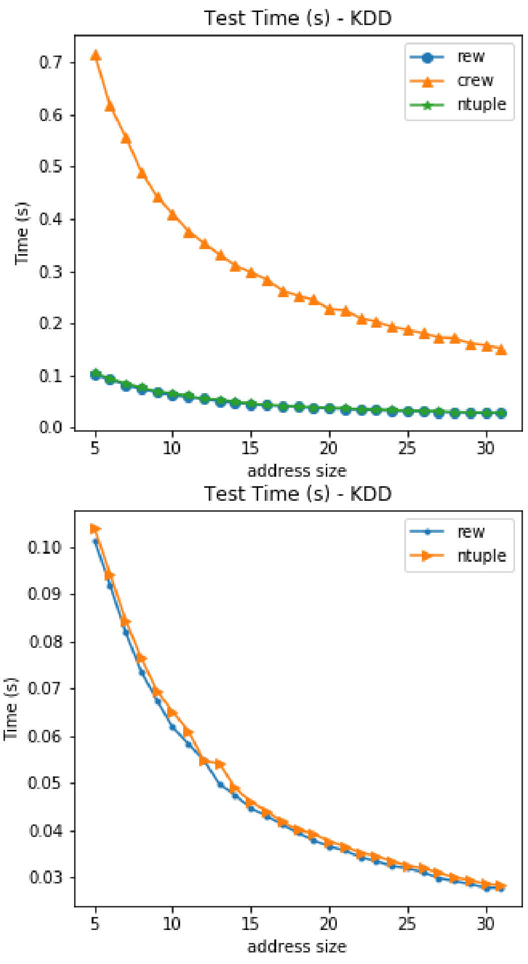


Figure 5.5: Address size X test time (s) for: (a) ReW, CReW and n-tuple Regression Network in KDD18 dataset; (b) ReW and n-tuple Regression Network in KDD18 dataset.

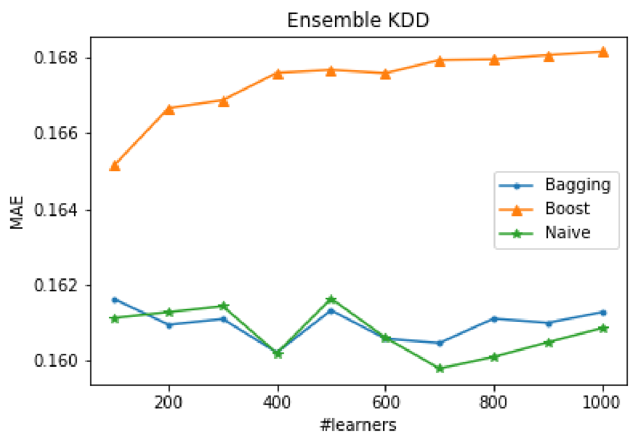


Figure 5.6: Number of weak learners X MAE for ReW Bagging, ReW Boosting and Naïve ReW Ensemble in KDD18.

KDD18 challenge.

One caveat: the n-tuple Regression Network used here shares the current implementation of dictionary-based WiSARD models where only memory locations accessed at some point in the training phase are allocated, causing the memory consumption of the model to be quite small. Experimental results show that, in general, ReW and CReW outperform n-tuple Regression Network in MAE, training, and test time.

Some considerations: when the address size n of a network increases, it becomes naturally more sparse, so the confidence of the network decreases; both ReW and CReW consistently present accuracy/MAE with low standard deviation; the output of an ensemble is obtained by the average output of all its members, using the same modalities used internally in the ReW model.

It can be seen from the results of Table 5.2 that both proposed models presented small differences from the state-of-the-art while surpassing its speed in many orders of magnitude. Another experiment carried out involved several ReW configurations using the KDD18 challenge winner preprocessing setup (ignoring categorical variables) and the best result was 0.09447 (thermometer = time window size = 10, n = 30, minZero = minOne = 0, harmonic power mean). Experiments with Naïve ReW ensembles introduced a slight improvement in the performance of the models, despite the natural drop in speed.

The best results from ReW Bagging and ReW Boosting are 0.13867 (40% of training set, 705 learners, harmonic mean, training time = 1.54s, test time = 3308.82s) and 0.14996 (905 learners, simple mean, training time = 5.16s, test time = 4689.84s), respectively. These ensembles have been found to have worse results than the individual models and were obviously much slower. This comes as no surprise since there is no evidence that an ensemble will outperform the best model on the committee, only that it will do so to the worst model. Additionally, since no pruning strategy was applied and only ReWs and CReWs were used, the great advantage of ensembles in using the diversity of models to improve the predictive power of the system may have been missed. In this sense, one possibility of increasing the accuracy of ReW Ensembles would be to use preprocessing or even different features for each learner, rather than just increasing the number of learners by varying their parameters. A hybrid ensemble of 45 ReWs (with different n, minZero, minOne and averages) and a GradientBoost (n estimators = 8000, max depth = 1, loss = lad, learning rate = 0.01) achieved 0.08814 in MAE.

5.3.3 Analysis of Experiments in Other Datasets

For a better analysis of the behavior of weightless regression models, they have been validated on datasets other than KDD18. The scale of values to be predicted is given in each of the datasets so that the MAE-based comparison makes sense, as this metric is related to absolute error. The datasets used here are:

- **House Prices[242]:** The most famous regression model benchmark, has 77 features (both categorical and numerical) and the challenge here is to predict the selling value of each home from its attributes (the training set has 973 examples and the test set has 480). Its y varies between 34.9k and 755k;

- **CalCOFI[243]:** This data set represents the longest and most complete time series of oceanographic and larval fish in the world. Although this dataset can be used for many different domains, here it was used to predict sea temperature from salinity level (training set: 579458, test set: 285405). Its y varies between 1.44 and 31.1;

- **Parkinson's Telemonitoring dataset[244]:** The purpose of this dataset is to predict for each patient their UPDRS, a continuous value on a scale that measures an individual's motor disorder level. 16 features are provided per patient (training set: 3936, test set: 1939). Its y varies between 0.00083 and 0.09999.

The choice of datasets was as follows: House Prices is a dataset traditionally used in benchmark of regression models, having many features and little data; CalCOFI is a dataset where the explanatory random variable is composed of a single feature and has a lot of data for training and testing; Parkinson's dataset is a balanced base regarding the number of features and data.

In these datasets, ReW, CReW and their ensembles were compared to the n-tuple Regression Network, GradientBoost, and XGBoost. The metrics collected in these experiments were Mean Absolute Error, the standard deviation for Mean Absolute Error, training, and test time. The models were validated 10 rounds each. For the experiments using weightless models, data preprocessing was done using one-hot encoding for categorical variables and a thermometer for numerical variables. The thermometer had its size varied from 5 to 30 bits and the tuple address size range was calculated according to the size of the thermometer (all values divisible by data word length from 2 bits were used). The results of these validations are shown in Fig. 5.7-5.21 and Table 5.3. Overall, the average type had little impact on model error.

About the models used in the benchmark:

- GradientBoost[245, 246]: a specific type of boosting algorithm, whose models are usually decision/regression trees, that generalizes their models by optimizing an arbitrary differentiable loss function;

- XGBoost[240]: an implementation of GradientBoosting that can be used by distributed processing frameworks and has become very popular because it has won many competitions within the machine learning community.

In the House Prices dataset, XGBoost and GradientBoost performed better, but the weightless models were competitive, especially ReW. ReW and n-tuple Regression Network were the fastest training models, followed by CReW, while XGBoost and GradientBoost achieved the worst performance. Regarding the test speed, XGBoost and GradientBoost had better performance compared to the weightless models, being CReW the slower model, due to successive draws during the classification it performs during the prediction. In the Parkinson dataset, XGBoost and GradientBoost had the smallest error, followed by ReW and CReW, which were still competitive. ReW and n-tuple Regression Network performed better on both training and test times, followed by CReW.

In the CalCOFI dataset, all weightless models performed equally in validation. This is because only one feature was used in these experiments and no thermometer setup made any significant changes to the resulting data words. CReW also did not create any ReW discriminators other than the original in these experiments. In both the training set and the test set, the weightless models outperformed GradientBoost, but had higher error than XGBooost. At training time, ReW and n-tuple Regression Network outperformed CReW, which in turn was faster than GradientBoost and XGBoost. At test time, the increasing order of performance was XGBoost, CReW, ReW, n-tuple Regression Network, and GradientBoost.

In all cases, when large-size thermometers were used for preprocessing, the new weightless models are competitive with XGBoost and GradientBoost. In general

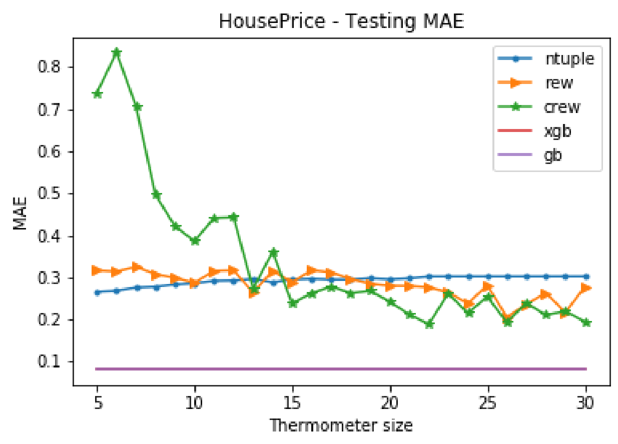


Figure 5.7: Thermometer size X MAE for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in House Prices.

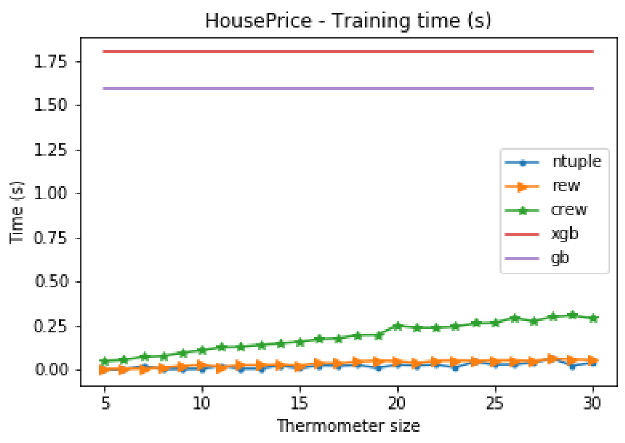


Figure 5.8: Thermometer size X training time(s) for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in House Prices.

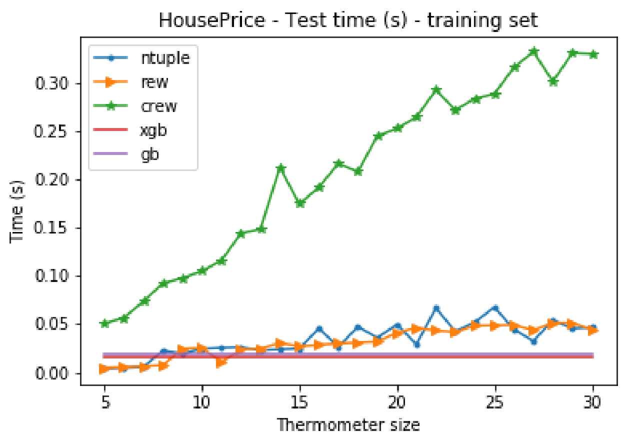


Figure 5.9: Thermometer size X test time(s) in training set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in House Prices.

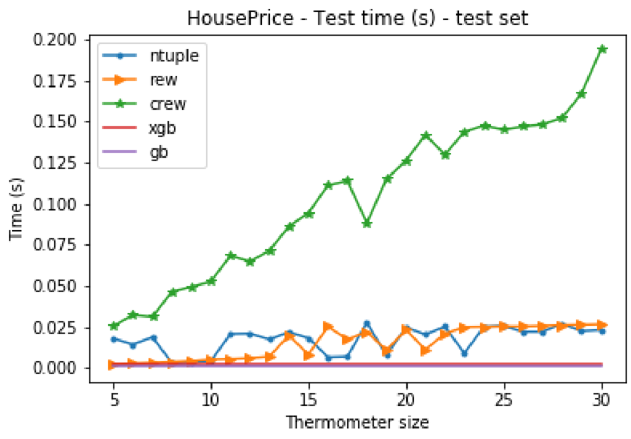


Figure 5.10: Thermometer size X test time(s) in test set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in House Prices.

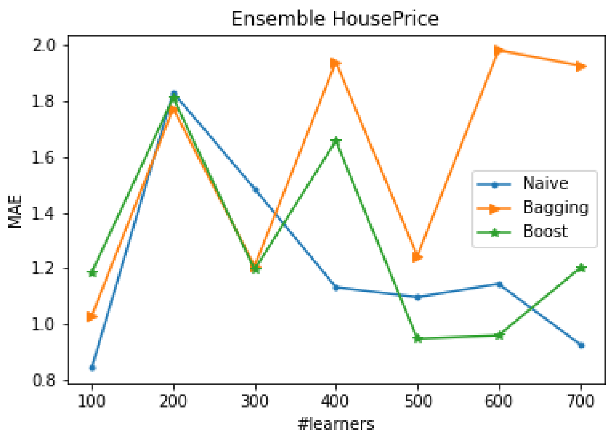


Figure 5.11: Number of weak learners X MAE for ReW Bagging, ReW Boosting and Naïve ReW Ensemble in House Prices.

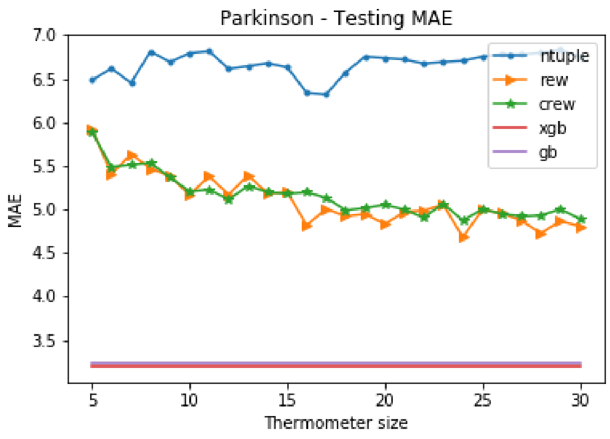


Figure 5.12: Thermometer size X MAE for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in Parkinson.

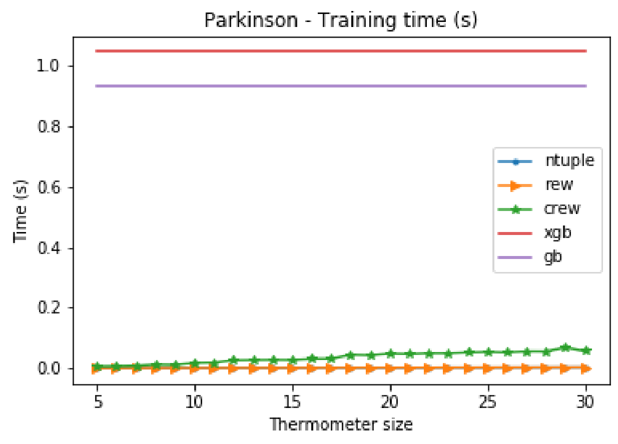


Figure 5.13: Thermometer size X training time(s) for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in Parkinson.

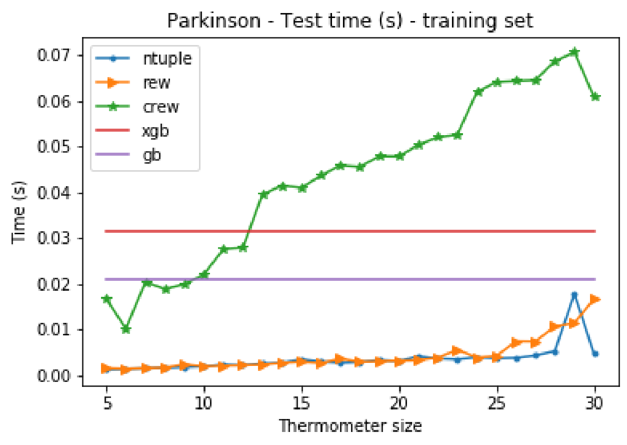


Figure 5.14: Thermometer size X test time(s) in training set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in Parkinson.

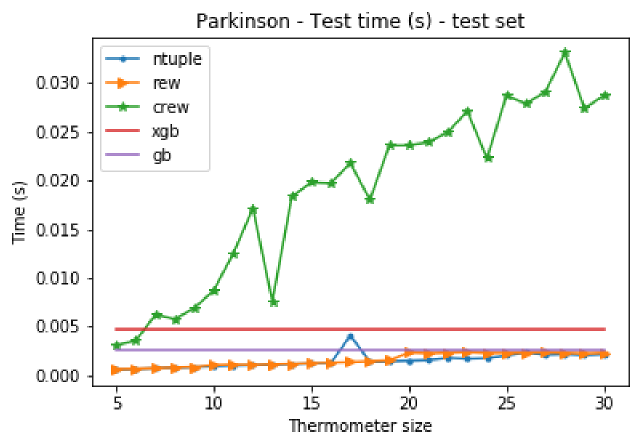


Figure 5.15: Thermometer size X test time(s) in test set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in Parkinson.

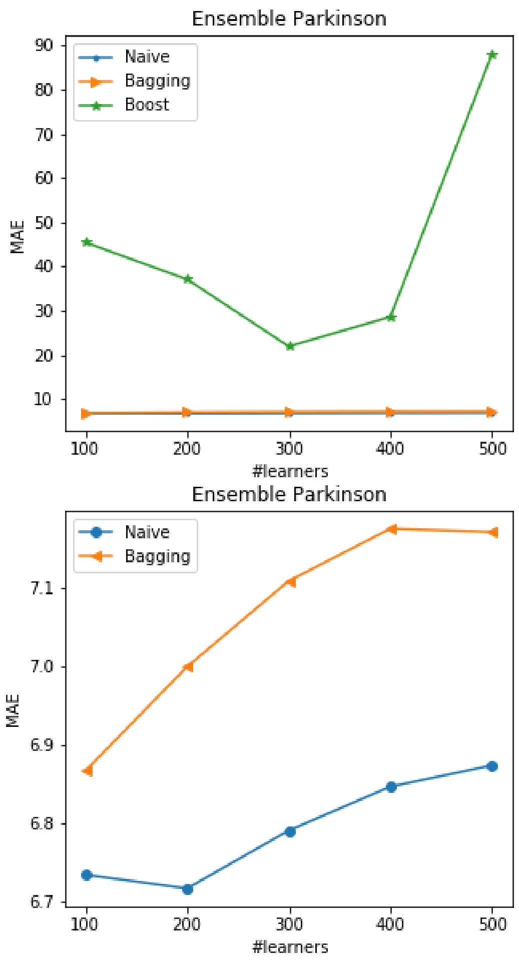


Figure 5.16: Number of weak learners X MAE for: (a) ReW Bagging, ReW Boosting and Naïve ReW Ensemble in Parkinson; (b) ReW Bagging and Naïve ReW Ensemble in Parkinson.

Table 5.3: Best results for weightless models with standard deviation and best median type. Caption: Md: median; PM: Power Mean; HM: Harmonic Mean; GM: Geometric Mean.

	House Prices	**Parkinson**	**CalCOFI**
ReW	0.278 ± 0 (Md)	$\mathbf{4.806 \pm 0}$ **(HM)**	$2.412 \pm 4.44 * e^{-16}$ **(PM)**
CReW	$\mathbf{0.194 \pm 0}$ **(PM)**	4.893 ± 0.105 (GM)	$2.412 \pm 4.44 * e{-}16$ **(PM)**
n-Tuple RN	0.302 ± 0	6.75 ± 0	$\mathbf{2.412 \pm 0}$

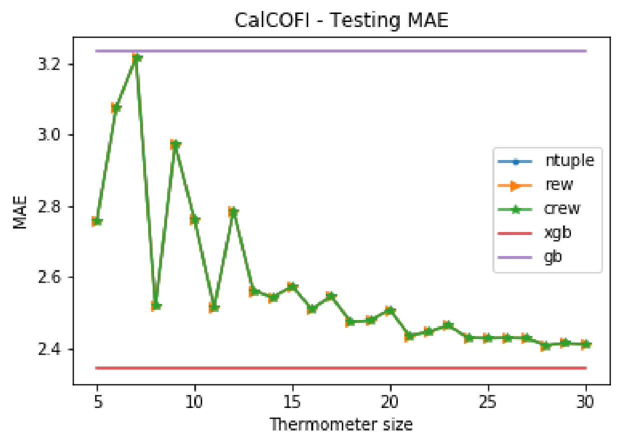


Figure 5.17: Thermometer size X MAE for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

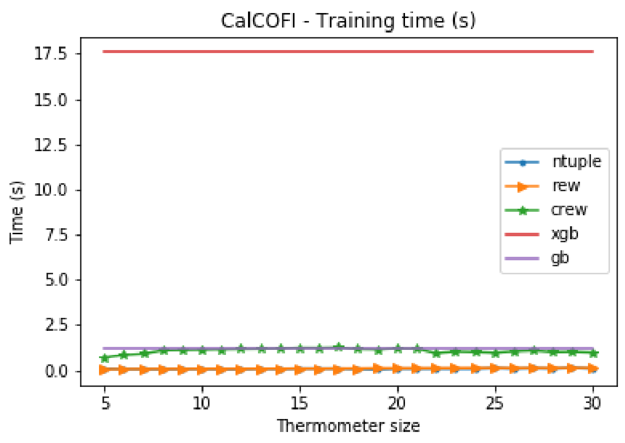


Figure 5.18: Thermometer size X training time(s) for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

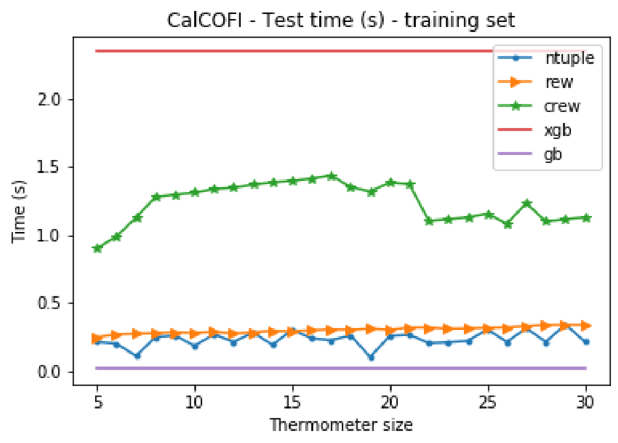


Figure 5.19: Thermometer size X test time(s) in training set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

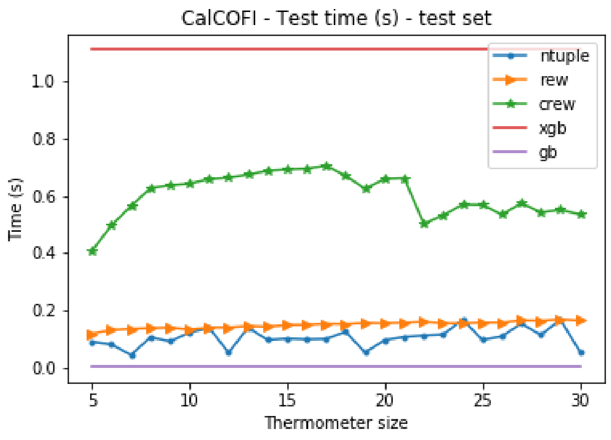


Figure 5.20: Thermometer size X test time(s) in test set for ReW, CReW, n-tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

ReW and CReW outperformed the n-tuple Regression Network, except for CalCOFI dataset, where the three models were completely equivalent.

5.3.4 Regression WiSARD's Learning Curves

Since one of the key features of WANN is precisely its training speed, this type of model becomes a strong candidate for tasks that require online learning, which necessarily implies that the model has to be able to generalize its learning from few examples. to provide an effective prediction for new examples.

ReW, CReW, and n-tuple Regression Network had their learning curves obtained from an experiment using the House Prices dataset, where at each iteration the model learned a new example of the training set and predicted the entire test set. The results presented here are the average of 10 rounds of experiments.

The learning curves for MAE are shown in Fig. 5.22, proving that ReW and CReW can perform well from a reduced training set, performing well with far fewer examples than the original model.

5.3.5 Analysis of Ensemble Composition

An analysis of the influence of the type of model that makes up an ensemble (ReW only, CReW only, or both models) was made through a comparison of the three different ensembles types in KDD18 dataset using a fixed size 28 bits thermometer and 500 learners each. Experiments were performed 10 rounds each, and the MAE, standard deviation, training and test time of the ensembles are laid out in Table 5.4.

Concerning MAE, the advantage of Naïve ReW Ensemble possibly came from the large training set learned by its models. Once no pruning techniques were employed in these ensembles, probably the smallest training set used in ReW Bagging and

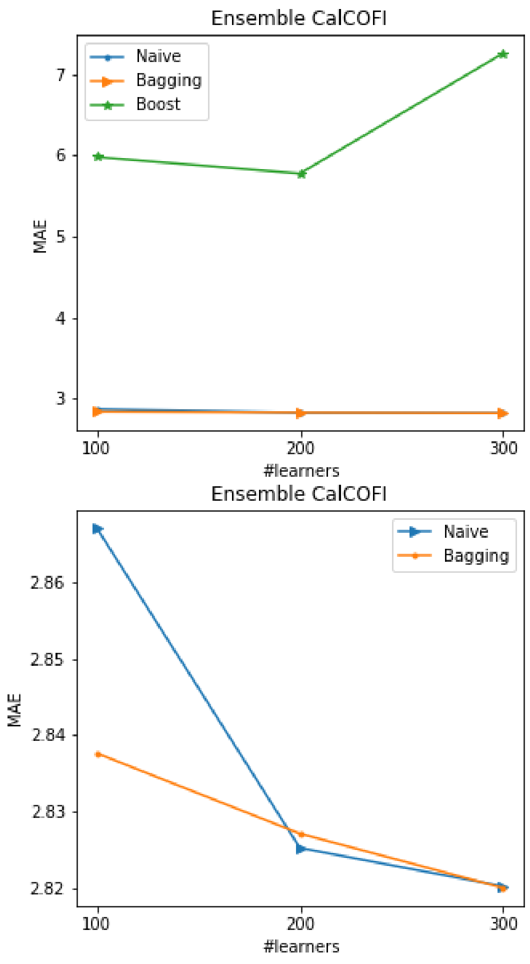


Figure 5.21: Number of weak learners X MAE for: (a) ReW Bagging, ReW Boosting and Naïve ReW Ensemble in CalCOFI; (b) ReW Bagging and Naïve ReW Ensemble in CalCOFI.

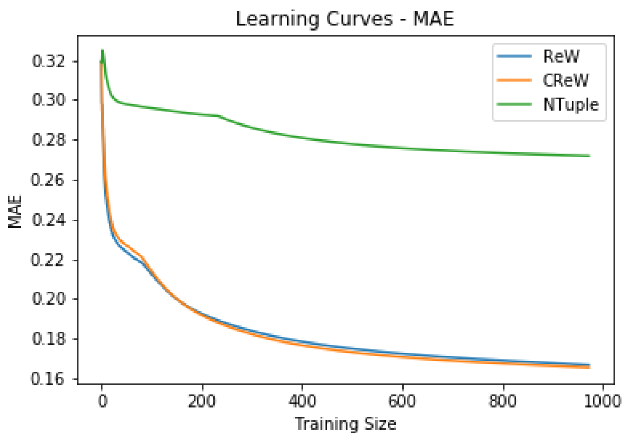


Figure 5.22: Length of training set X MAE for ReW, CReW and n-tuple Regression Network in House Prices.

Table 5.4: Comparison of the three types of ensembles using only ReW, only CReW and both models. Caption: MAE: mean absolute error; TrT: training time (s); TT: test time (s)

Type of Ensemble	Metric	Only ReW	Only CReW	Mix
Bagging ReW	MAE	$0.162 \pm 5.92 \times 10^{-4}$	$\mathbf{0.16} \pm 4.94 \times 10^{-4}$	$0.16 \pm 1.93 \times 10^{-3}$
	TrT	**19.93 ±2.23**	92.07 ± 8.09	58.14 ± 5.73
	TT	**4831.29 ± 201.64**	4667.07 ± 1386.91	4571.79 ± 1326.17
Boost ReW	MAE	$0.168 \pm 2.26 \times 10^{-4}$	$0.168 \pm 4.14 \times 10^{-4}$	$\mathbf{0.168} \pm 2.17 \times 10^{-4}$
	TrT	**2.12 ± 0.27**	9.84 ± 1.16	5.93 ± 0.7
	TT	**4902.03 ± 192.67**	5136.76 ± 459.88	5043.16 ± 261.35
Naïve ReW Ensemble	MAE	$0.162 \pm 4.21 \times 10^{-4}$	$\mathbf{0.161} \pm 4.62 \times 10^{-4}$	$0.162 \pm 7.54 \times 10^{-4}$
	TrT	**26.74 ± 1.75**	121.42 ± 11.83	74.01 ± 11.24
	TT	**5034.41 ± 174.12**	5241.58 ± 501.5	4504.95 ± 1633.41

Boosting do not turn the weak learners into specialists in specific features but just networks with less training. In most of the cases Naïve ReW Ensemble outperforms ReW Bagging, but both ensembles are competitive. In these cases, ReW Bagging has as advantage less computational cost and training/test speed due to the less training of the weak learners, which implies in test speed due to the number of turns in bleaching.

These data show that the increasing order of training speed is ReW Boosting, ReW Bagging, and Naïve ReW Ensemble, which is intuitive as this is the order in which the size of training sets increases. The time to validate and determine the weight of each weak learner's vote has made ReW Boosting the slowest, yet least trained ensemble.

The testing speed of ReW Bagging and Naïve ReW Ensemble was equivalent since the procedure for prediction of these ensembles is equal. ReW Boosting was slightly slower in prediction, due to the calculation of each learner's vote value based on the weights obtained in validation.

As for the choice of models that make up the ensemble, ReW, CReW, and both at the same time obtained equivalent MAE. Ensembles with only ReW were obviously faster in training and prediction, as CReW performs a classification step in both of these phases. As expected, mixed ensembles had intermediate performance between homogeneous ReW and CReW ensembles since they had both types of models.

5.3.6 Logistic Regression

Using ReW's multi-dimensional RAM, an extension of the model was made to deal with logistic regression, storing probabilistic values in the y dimension and using the logistic function as a mean.

5.3.6.1 The Model

Regression WiSARD's additions to the original model include the ability to perform logistic regression, that is, to estimate the probability associated with the occurrence of a given event in the face of a set of explanatory variables.

Logistic regression is equivalent to separating events in a plane with a linear boundary, which could be represented by the equation $\beta_1 x + \beta_2 y + c = 0$. Therefore, any other point (a, b) can have its position in relation to the linear boundary expressed by $E(a, b) = \beta_1 a + \beta_2 b + \gamma$. To ensure that this function is limited to the range (0, 1), the logistic function ($g(x) = \frac{1}{1+e^{-x}}$) is applied to it.

Therefore, it has as hypothesis:

$$H(x, y) = g(E(x, y)) = \frac{1}{1 + (e^{-(\beta_1 x + \beta_2 y) + \gamma})} \tag{5.1}$$

Of course, a dataset with binary events (may or may not have happened) will have annotations on the status of each event. If you want to calculate the probability of an event occurring using the WiSARD Regression, simply train them with the dataset examples, using $y = 0$ for no occurrence of the event and $y = 1$ if it occurred. Then it is enough to predict the event that it is desired to calculate the probability, using as average the logistic hypothesis, where x will be the sum of the counters, y will be the number of memory locations accessed with $sum(y)$ not null, β_1 will be the ratio between examples learned and those where the event actually has occurred, β_2 is the degree of activation of the RAMs and γ is a hyper-parameter which must be calibrated.

5.3.6.2 Validation

To validate ReW's logistic regression was used the Criteo dataset [247], a database where each sample represents the behavior of a person browsing a site, and the event whose probability is to be estimated is that of clicking on an advertisement link on that page. The dataset has 24 days of click logs, where each day has 18 million events (48 GB of data). Each sample has 39 features (13 continuous and 26 categorical), all of which are anonymous. This is a very unbalanced classification problem (class "did not click" has 52% more occurrences than class "clicked").

For this test, only 1.8 million events were used, 90% for training and 10% for testing. The categorical variables were not used and the continuous variables were preprocessed with a binary thermometer. The estimated probability for the click was validated using logarithmic loss:

$$logloss = -(y\ log(p) + (1 - y)\ log(1 - p)), \tag{5.2}$$

where y will be 0 when the event has not occurred and 1 case has occurred and p is the probability estimated for its occurrence.

The lower the logarithmic loss, the better the prediction. Were used ReW with n = [5,15,25,35] and all obtained log loss = 0.493. The training and test time of each network is indicated in Table 5.5.

Table 5.5: Time spent by ReW to perform logistic regression on the test set

Address size	Training time (s)	Prediction time (s)
5	**2.19**	**1.10**
15	6.89	3.23
25	11.43	5.42
35	15.79	7.27

5.3.6.3 Discussion

In practice, this demonstrates how ReW can be used in other domains, where it needs to approximate functions that are not necessarily non-parametric, just using an appropriate estimator and, perhaps, using a different kernel. This is one more contribution that this model offers in reaction to the original n-Tuple Regression Network. In this case, the kernel estimator used was the logistic regression equation instead of the traditional Nadaraya-Watson estimator, while the kernel $(\boldsymbol{x} - \boldsymbol{x^i})$ was kept.

5.4 Chapter Conclusion

This chapter presented two new weightless neural networks for regression tasks based on the n-tuple Regression Network model, both competitive in terms of state-of-the-art accuracy and other results relevant to the prediction problem of productivity of palm oil of the KDD18 competition. These results were published in [2, 195]. In terms of learning and prediction times, the two weightless models proved to be superior to all other solutions and, due to their simplicity, these networks are ideal candidates for situations that require online learning and low computational costs.

This work also explores the use of three types of weightless neural networks regression ensemble: Naïve ReW Ensemble, ReW Bagging, and ReW Boosting, with Naïve ReW Ensemble achieving better performance than the others. An exploration of the minZero and minOne thresholds was also done for the weightless regressors and their impact, mainly on ensembles, was demonstrated by the results.

Other works that used ReW in other contexts include an agnostic classifier interpreter[248] and a controller on DC STATCOM converter under fault conditions[249].

An interesting legacy of Regression WiSARD consists of its multi-dimensional RAM nodes, where each memory location stores not only a counter but an array of values. In the models proposed here, the content of this array is reduced to just one counter and the sum of the prediction values associated with the examples learned by the model, however, many other values could be added, according to the problem domain. This type of RAM can be used for both classification and regression tasks, in addition to more complex tasks, such as reinforcement learning.

ReW with other regression models, such as deep neural networks. Other metrics must be used in this benchmark since this work used only MAE and MSE, which measures the absolute magnitude of errors. Other metrics, such as Mean Error (which measures the additive bias in the error), Mean Squared Log (useful when dealing with right-skewed targets), Median Absolute Deviation (standard deviation robust to outliers), and coefficient of determination R^2 (a measure of the ratio of variability that the model can capture vs the natural variability in the target variable) need to be used in validations involving ReW and other models so that a fairer and more complete comparison is made and the strengths and weaknesses of the weightless regressors are more evidenced.

Ongoing and future research include: (i) adding new policies to update *sum(y)*; (ii) exploring the possibility of different address sizes inside CReW discriminators, with new decision policies adapted to different neurons (since each discriminator will be trained with a sub-profile of the data due to their similarity to each other while distancing themselves from the samples learned by other discriminators, it may be interesting to try to estimate the *pdf* for each data group using a smoothing factor distinct, or even a distinct estimator and kernel); (iii) varying the types of preprocessing and features used in the ensembles; (iv) exploring the Logistic WiSARD in different scenarios; (v) adding strategies for ensemble pruning; (vi) apply ReW in auto-regression tasks and (vii) change the kernel used in the prediction.

Chapter 6

A Weightless Multi-modal Empathy Predictor

The ability to experience and evaluate the degree of empathy is a vital skill for human survival and the construction of its social interactions. Empathy prediction is, consequently, an area of great interest within affective computing, since many other computational tools can make a great profit from it, for instance, virtual tutors, security systems, and dynamic customization of applications oriented to the detection of instant emotions.

A usual way of categorizing emotions is through valence and arousal attributes, both assuming continuous values. Valence refers to how pleasant a particular feeling was (a negative valence refers to an unpleasant event) and arousal to how intense it was. A slightly happy situation would have a positive and high valence, while it would tend to have arousal close to zero. Thus, a trend in affective computing is to predict each of these attributes separately in tasks of emotional analysis.

The prediction of those values becomes even more complex considering real-world scenarios, in which multiple subjects are interacting with each other simultaneously and with concomitant noisy information provided by the background. Moreover, there are different levels of emotive signals that provide an additional layer of complexity, e.g., facial information, the body language of the individuals, the sound spectrum of the conversation, and possible semantic content of the dialogues. All this amount of complexity makes it difficult to create highly accurate affective prediction systems that operate in real-time.

Aiming to offer an approach that is capable of providing fast and effective responses, in addition to having the ability to learn online, this work presents an affective valence prediction system based on weightless neural networks. This system is validated on the OMG-Empathy Prediction dataset[5], which consists of dialogues on a wide range of previously chosen topics.

This chapter has a highly experimental bias, without dwelling on an elaborate

Table 6.1: A comparison between the multi-modal solutions in literature.

	Video mode	**Audio mode**	**Text mode**	**Fusion**
[250]	VGGFace2-based CNN and VGG FERCNN	PANN-based DNNs and 1DCNN + LSTM-based DNNs	-	Weighted fusion score
[251]	VGG16 CN	ELECTRA	TRILL	Transformer
[252]	DNN	DNN	DNN	Multi-modal DNN-based embedding generator

theoretical exploration of multi-modal learning. The main purpose of this work was to validate the potential of Regression WiSARD in an open and complex task, whose solution is still being explored by the community.

6.1 Related Work

Multimodal learning is an area that has received increasing attention, given the increasing complexity of real-world problems. Some work related to video: Dresvyanskiy et al.[250] uses VGGFace2-based CNN, VGG-FERCNN-based video features combined with PANN-based-deep networks and 1DCNN + LSTM-based deep networks audio features through weighted fusion scores.

Yu et al.[251] incorporates frame features extracted from VGG16 CN, text features from ELECTRA, and audio features from TRILL. After feature extraction, the time-series features, frames and audio are fed into Transformer encoder layers to return a vector representing their corresponding modality. Frames, title, description, audios features pass through a context gating layer before concatenation. Then a vector in the latent space is generated and select the frame that is most similar to this vector in the latent space.

Moon et al.[252] proposes a multimodal deep learning framework that can transfer the knowledge obtained from a single-modal neural network to a network with a different modality. This approach learns the analogy-preserving embeddings between the abstract representations learned from each network, allowing for semantics-level transfer or reconstruction of the data among different modalities. This method is thus specifically useful when one of the modalities is more scarce in labeled data than other modalities. This comparison is better visualized in Table 6.1.

6.2 Empathy Prediction

A classic issue of affective computing involves the attainment and use of spontaneous emotions. In the study of emotive valence, the degree of positivity or desirability of an affective state, spontaneity, becomes even more indispensable. Measuring valence in the dialogue between an actor and a listener, vaguely scripted, is the basis of the OMG-Empathy Prediction Challenge.

The dataset is composed of 80 videos, each lasting approximately 5 minutes, in which actors tell stories to listeners and adapt their performance according to the emotions expressed by the listeners, who then evaluated their valence level with a continuous value between -1 and 1 at each point in the conversation. A example of OMG video is displayed in Figure 6.1.

The OMG-Empathy Prediction dataset is divided into training, validation, and test (with 4, 1, and 3 stories, respectively). The challenge is separated into two tracks: Personalized Empathy and Generalized Empathy. In the former, the goal is to predict the empathy of the individual subjects and in the latter the empathy of the conversation itself. During the challenge, we use the validation dataset to find the best combination of hyper-parameters for the model for the test set. However, once the annotated test set was released, we looked for the best parameters in it directly. Here you can find the results of the experiments in both data sets, but it should be noted that the test set had a particular hyper-parameter exploration.

The metric for assessing the proximity between predicted values and those provided by subjects is the concordance correlation coefficient (CCC) ρ_c, which measures the agreement between two variables, x and y. CCC is given by

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}, \tag{6.1}$$

where μ_x and μ_y are the means of x and y, σ_x and σ_y their standard deviations, and ρ is the Pearson's correlation coefficient between both variables.

6.3 Experimental Results

6.3.1 Experimental Setup

The proposed architecture uses the pipeline described below.

- **Preprocessing**: this step consists of preprocessing each frame and obtaining a binary input for the networks. Two preprocessing procedures occur here: I. the binarization of the subject face in the Personalized Track and the face of both individuals in the Generalized Track, being obtained from the facial

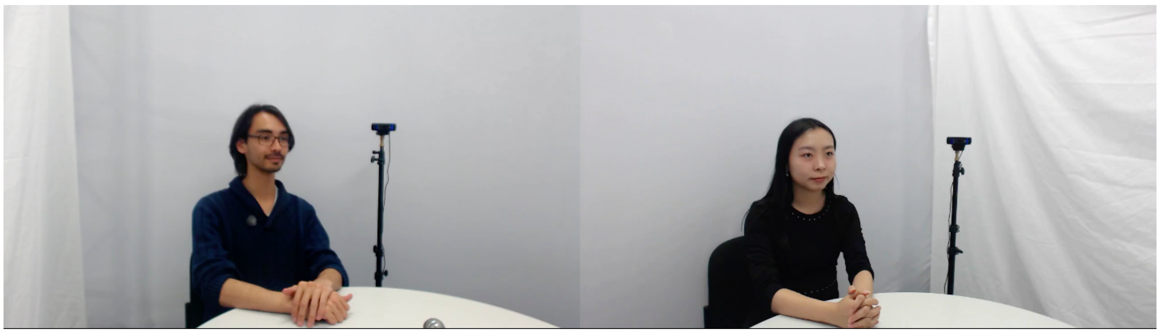


Figure 6.1: Example of the Sub Track Empathy prediction of OMG-Empathy datase[5].

extraction script provided by the challenge, and II. the extraction of features from the audio contained in the interval that spans from 30 seconds before to 30 seconds after a given frame being treated. These two inputs are then concatenated. This pipeline is illustrated in Figure 6.2;

- **Storage**: because preprocessing is the slowest and most costly phase of the entire validation process, it was opted to store the frames of a given story that were are preprocessed in structures called ReW datasets (a type of dictionary with inputs and their respective predictions). This would make it simpler for that preprocessed data to be reused in tests with different network configurations. Each ReW dataset contains a complete story for a combination of one type of image preprocessing and one of audio, using a single combination of hyperparameters;

- **Validation**: a given network (ReW or CReW) receives training and test ReW datasets, which are employed to perform the desired prediction. The same network can use ReW datasets of the same story with distinct preprocessing methods at the cost of treating the difference in the size of its inputs. If an input is smaller than the retina, it is padded with 0s until the appropriate size. If it is larger, it will be staggered by joining its bits with a logical 'or', that is, given two bits this operation returns that with the highest value.

The experiments were performed using single instances of ReWs and CReWs, addressing n-tuples ranging in length from 5 to 30 bits. The experimental environment was an Intel Core i7-6700, CPU @ 3.40GHz with 4 cores, 32GB of RAM, and GeForce GTX 960. All experiments were run just one time due to the dataset's size and the preprocessing time.

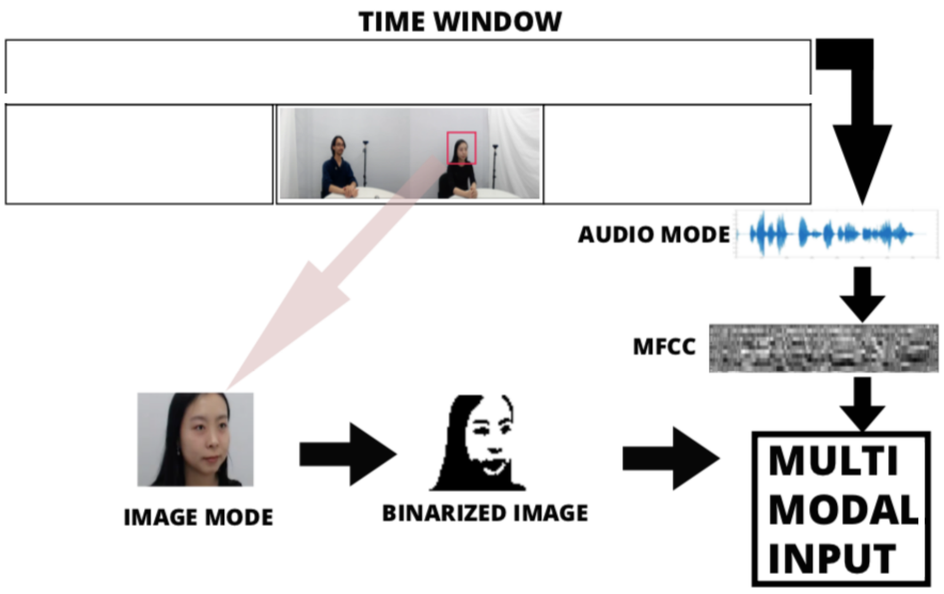


Figure 6.2: WiSARD's multi-modal pipeline

6.3.2 Results in Validation Set

This section will describe the results of different types of networks and preprocessing methods employed on the Personalized Track in Validation Set.

Initially, a validation of the proposed architecture was performed by using 90% of the training base and subjects 1, 2, 3, 4, 6, 7, 8, and 9 of the validation base (story 1), with Regression WiSARD and ClusRegression WiSARD using only Sauvola and MFCC preprocessing methods. Table 6.2 shows the average time of each preprocessing method.

Because of the dimensions of the dataset, it was split into several parts, which were then preprocessed simultaneously. Once already binarized, the dataset was stored in two RegressionDatasets, one for training and the other for validation. The networks employed for this had to address n-tuples whose length ranged from 2 to 30 bits. It is notable that despite the preprocessing time is high since all the datasets had already been binarized, it was a low cost one could pay to explore the whole space of hyperparameters of the network. Table 6.3 shows the values of CCC for each story given the prediction performed by ReWs with different addressing n-tuple lengths, with the best CCC for each subject properly marked in bold. Analogously, Table 6.4 presents the CCC for CReWs. Figures 6.3-6.6 depicts the variation of the mean CCC, correlation coefficient (ρ), training time and validation time for different addresses of ReW using only image mode. Figure 6.7 displays the CCC for ReW and CReW using both image and audio modes.

Tables 6.3 and 6.4 show the results in the validation set for ReW and CReW, respectively. The results of the ensembles in the Validation set are in Table 6.5,

Preprocessing	Average time (s)
Sauvola	$4*10^{-4}$
Canny	$2*10^{-4}$
Gaussian	$1*10^{-3}$
Otsu	$5*10^{-5}$
MFCC	10.54

Table 6.2: Average time of each preprocessing method.

	n					
Subject	5	10	15	20	25	30
2	0.09	0.08	0.04	-0.28	**0.6**	0.34
3	-0.02	-0.01	0.04	0.17	0.36	**0.38**
4	-0.25	-0.19	-0.07	0.19	0.44	**0.45**
6	0.13	0.12	0.18	0.24	**0.26**	0.2
7	0.21	**0.25**	0.3	-0.05	0.08	0.04
8	0.51	-0.35	0.06	0.26	**0.31**	0.25
9	0	0	0	-0.01	-3.16	**0.06**

Table 6.3: CCC for each story with different addressing values with ReW model.

	n					
Subject	5	10	15	20	25	30
2	0.09	0.08	-0.06	-0.29	**0.61**	0.35
3	-0.02	-0.01	0.11	0.17	0.35	**0.38**
4	-0.2	-0.19	0.08	0.19	0.44	**0.45**
6	0.09	0.12	0.22	0.24	**0.25**	0.2
7	**0.26**	0.25	-1	-0.06	0.07	0.03
8	-0.31	-0.37	0.24	0.27	**0.31**	0.25
9	0	0	-0.01	-0.02	-1	**0.06**

Table 6.4: CCC for each story with different addressing values with CReW model.

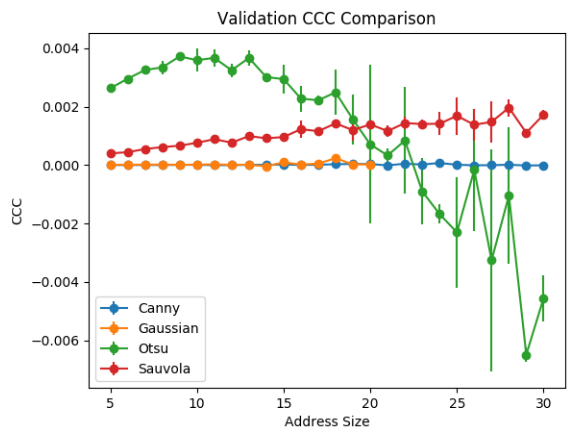


Figure 6.3: n X CCC for ReW in the validation set using only image mode.

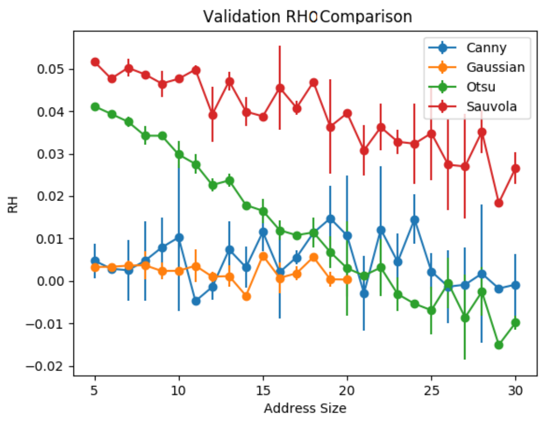


Figure 6.4: n X correlation coefficient for ReW in the validation set using only image mode.

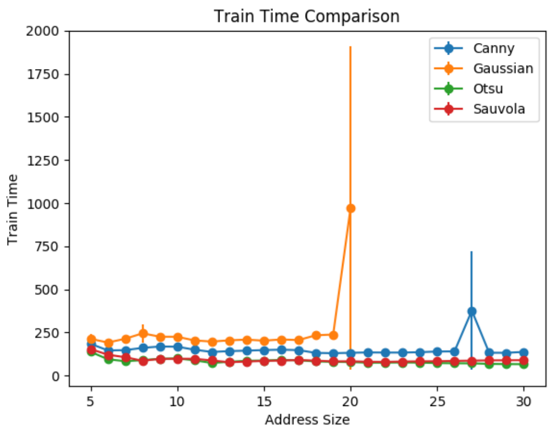


Figure 6.5: n X training time for ReW using only image mode.

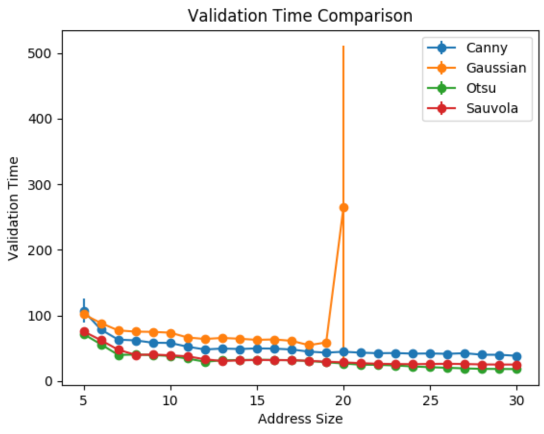


Figure 6.6: n X validation time for ReW in the validation set using only image mode.

all ensembles achieved CCC tending to zero, regardless of the preprocessing used. Bagging ReW was also tested in the test set, with virtually identical results.

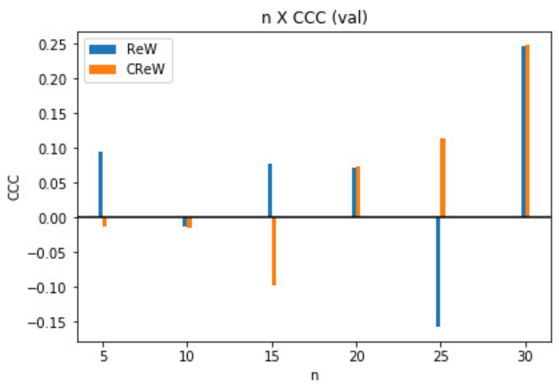


Figure 6.7: n X CCC for ReW and CReW in the validation set using image and audio modes.

Ensemble	**Binarization**	**Training (s)**	**Validation (s)**	**CCC**
Bagging	Gaussian	1.56	7523.65	0
	Sauvola	1.31	7396.88	9.93*e-05
	Canny	2.23	7481.77	-0.0003
	Otsu	1.15	7453.75	0.0001
Naïve	Gaussian	3.13	7571.90	0
	Sauvola	2.99	7678.32	-1.93*e-05
	Canny	1.34	7421.09	9.73*e-06
	Otsu	2.7	7459.93	-6.19*e-06

Table 6.5: CCC for each BaggingReW and Naïve ReW ensemble in Validation set of Personalized Track

6.3.3 Results in Test Set

Next, new experiments were performed using the entire training base and the entire test base, together with all combinations of image and audio preprocessing procedures. The mean CCC, correlation coefficient and prediction time for different network addresses of ReW are presented in Figures 6.8-6.10 using only image mode. Additional tests were realized using both image and audio modes and training plus validation sets for training the models. These results are displayed in Figures 6.11 - 6.20.

The comparison with the models submitted to the challenge that originated this domain is shown in Table 7.5. Except for the baseline, no model has its details available. This leaderboard is a partial representation of the one officially made available by the curators of the dataset[253].

6.3.4 Preprocessing Analysis

Concerning the preprocessing methods that were employed, it can be inferred from experimental results that:

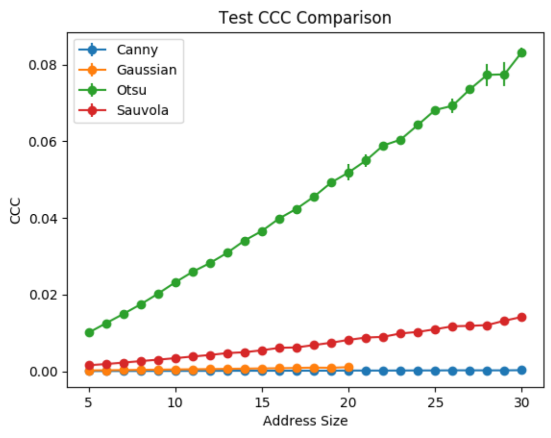


Figure 6.8: n (address size) X CCC for ReW in the test set using only image mode.

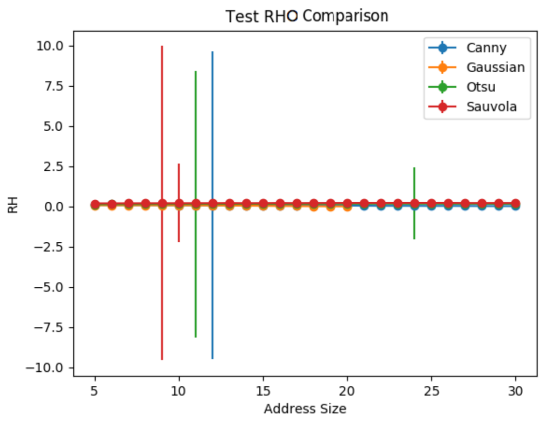


Figure 6.9: n (address size) X correlation coefficient for ReW in the test set using only image mode.

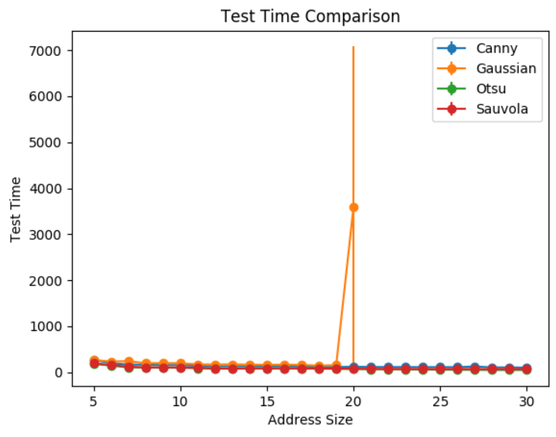


Figure 6.10: n (address size) X test time for ReW in the test set using only image mode.

- Otsu performed much faster than other methods, both in training and in testing, due to the reduced number of steps of its algorithm;

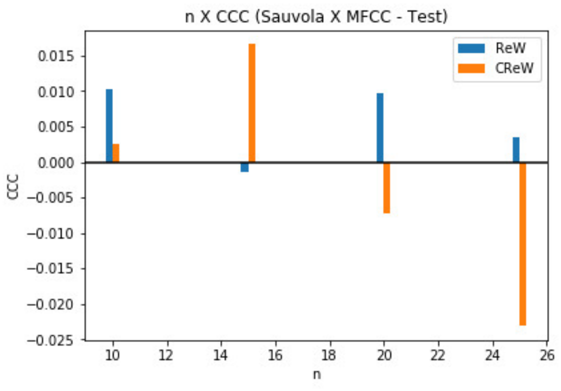


Figure 6.11: n (address size) X CCC for ReW and CReW in the test set (Sauvola method) using image and audio modes.

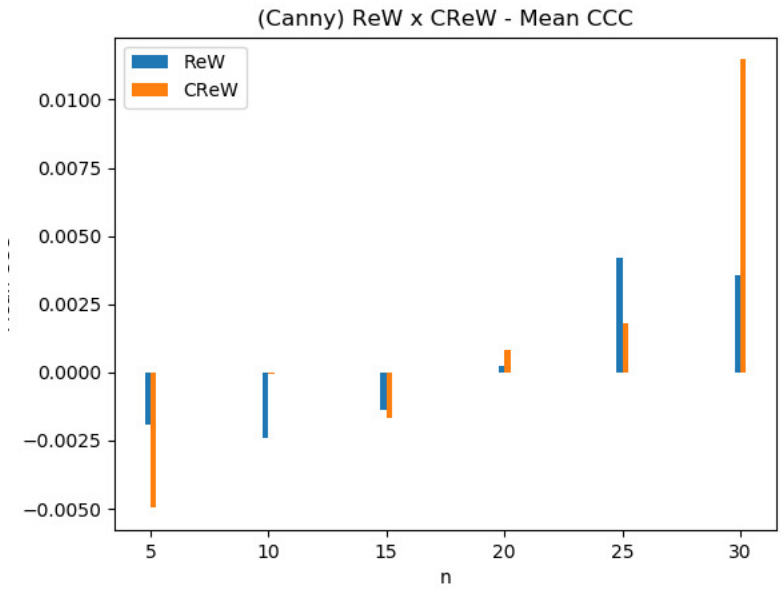


Figure 6.12: n (address size) X CCC for ReW and CReW in the test set (Canny Filter using image and audio modes.

- Gaussian adaptive filter and Otsu's binarization had their performance over time increased considerably as n-tuple size increased;
- ReW's performance degenerated faster than that of CReW with Otsu's binarization;
- ReW's performance increases faster as n-tuple size increases,
- Canny filter performed considerably better over test time than training time; and
- Sauvola method performed slower than the other preprocessing schemes in all cases.

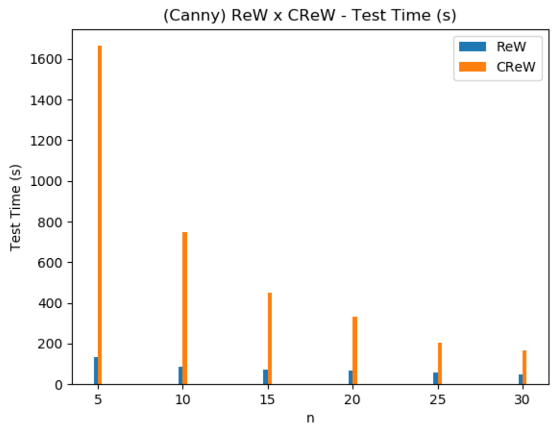


Figure 6.13: n (address size) X training time for ReW and CReW in the test set (Canny Filter) using image and audio modes.

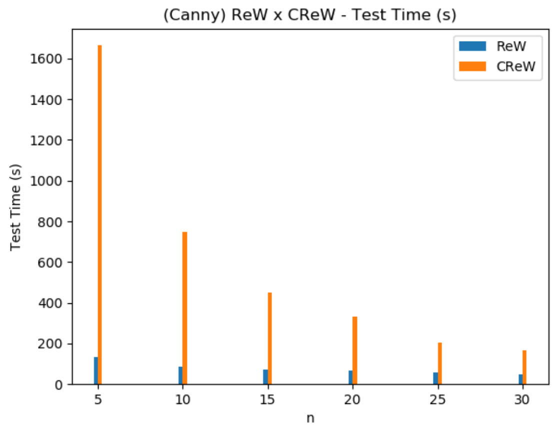


Figure 6.14: n (address size) X test time for ReW and CReW in the test set (Canny Filter) using image and audio modes.

6.3.5 Discussion

Some observations can be made from the analysis of the collected data: (i) in the validation set, where the networks are trained with a single story, although with distinct subjects, CCC tends to oscillate as the size of the n-tuple increases, reaching its apex for the more specializing configuration (n-tuple with the greatest length) for both ReW and CReW; (ii) in the test set, where the networks are trained with several stories, CCC tends to oscillate as n-tuple size increases for the ReW, and tends to decay for the CReW; (iii) training and prediction times decrease as n-tuple size increases, both for ReW and CReW, because as n-tuple size increases the number of RAMs decreases, and training and prediction times are directly proportional to the number of RAMs of the model; and (iv) CReW performance is always slower than that of ReW, because both in training and in prediction it needs to employ classification to determine which discriminator to use.

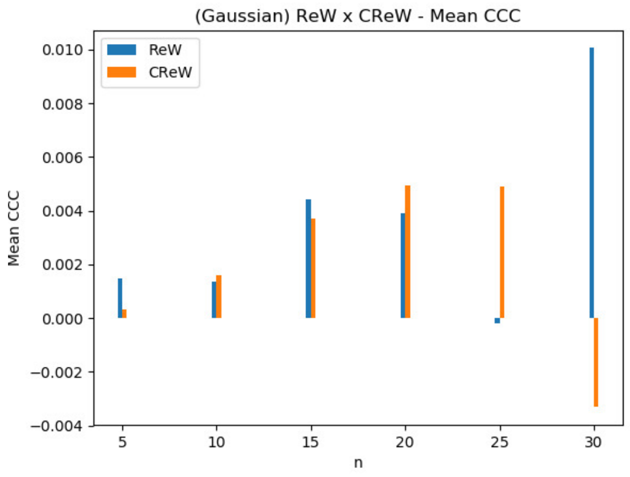


Figure 6.15: n (address size) X CCC for ReW and CReW in the test set (Adaptive Gaussian Filter) using image and audio modes.

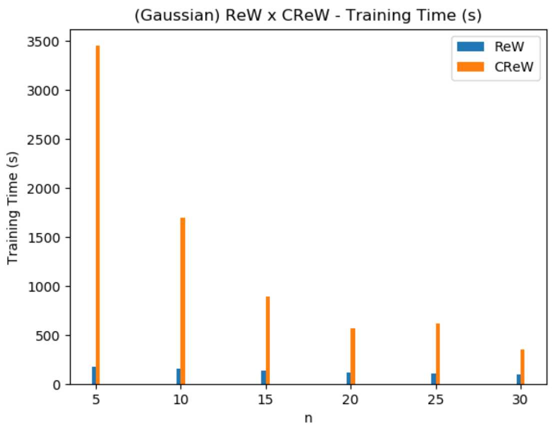


Figure 6.16: n (address size) X training time for ReW and CReW in the test set (Adaptive Gaussian Filter) using image and audio modes.

It can be concluded that the discrepancy of the behavior of the networks in validation and test sets are associated with the diversity of stories used for training. The variety of facial expressions and audio spectra resulting from this multiplicity of narratives might have produced similar inputs for different valencies, leading to the training of multiple distant values in the same memory positions, making a particular RAM contribute in the prediction phase with a value that does not genuinely correspond to the whole scope of valences related to the features trained there. So, this issue indicates that it would be required a system in which there are multiple networks, each one focused on a specific emotion and/or behavior. This would potentially avoid the overlapping of distinct features in the networks, thus enhancing its performance.

Individual neural networks outperform their ensembles in validation, probably

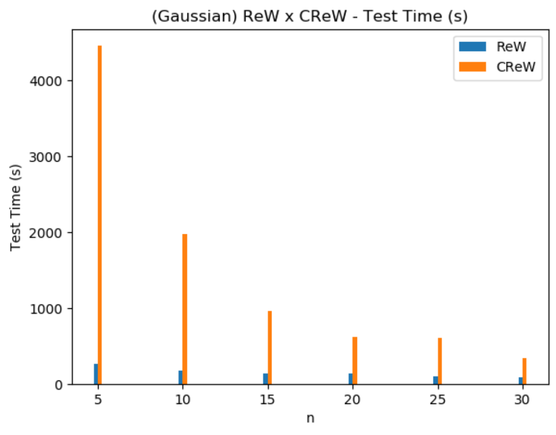


Figure 6.17: n (address size) X test time for ReW and CReW in the test set (Adaptive Gaussian Filter) using image and audio modes.

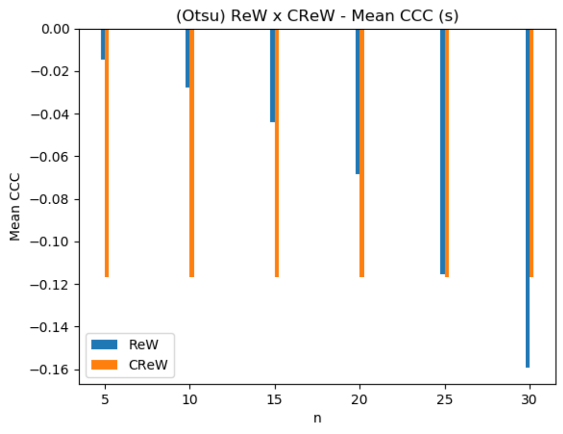


Figure 6.18: n (address size) X CCC for ReW and CReW in the test set (Otsu Binarization using image and audio modes.

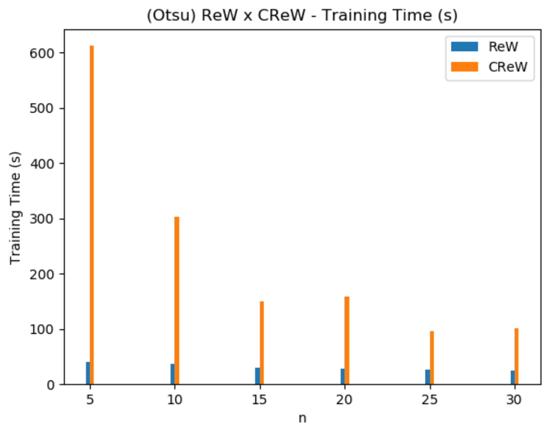


Figure 6.19: n (address size) X training time for ReW and CReW in the test set (Otsu Binarization) using image and audio modes.

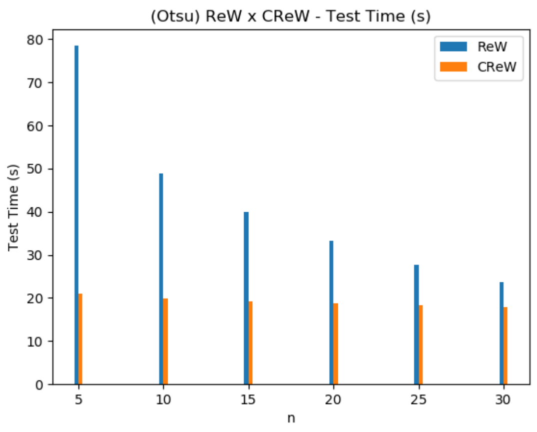


Figure 6.20: n (address size) X test time for ReW and CReW in the test set (Otsu Binarization) using image and audio modes.

Table 6.6: A general benchmark in OMG-Empathy Prediction dataset

Team	**Model**	**CCC**	**Modality**
Alpha-City	Manual	**0.17**	Audio + Image + Text
USTC-AC	Unknown	0.14	Audio + Image + Time
A*STAR AI	LSTM	0.14	Audio + Text
Alpha-City	Filters	0.12	Audio + Image + Text
Rosie	SVM	0.08	Audio + Image + Semantic
WANN	ReW	**0.08**	Image
Rosie	NN	0.07	Audio + Image + Semantic
A*STAR AI	LSTM	0.07	Text
Baseline	DNN	0.06	Audio + Image
Affective Bulls	Unknown	0.02	Audio + Image
WANN	ReW	0.01	Audio + Image

because the number of weak learners was not sufficiently representative. The lack of bootstrap and pruning algorithms for Bagging ReW probably caused the reduced learning of each learner to generate noise propagated by the ensemble, instead of efficiently combining the specialized learning of each weak learner. However, ReW and CReW used multimodal learning, while the ensembles used information provided only by the face image, so we see here the impact of adding more modes in this type of task.

6.4 Chapter Conclusion

This chapter explores the use of weightless neural networks in an empathy prediction task. These results were partially published[131]. Its major contribution is the unprecedented use of weightless regressors in a multimodal environment and the pioneering exploration of RAM-based models for the prediction of affective valences.

This task is an open question for the affective computing community, is considered a hard task, especially for in-the-wild settings. Given such a scenario, one could consider that weightless nets performed adequately for this task, as demonstrated in the validation set, where ReW obtained a CCC of 0.2 and had more than half the possible agreement for some subjects. In the test set ReW achieved 0.08 of CCC, being very competitive with the state-of-the-art, which obtained CCC 0.17 with a much more complex model. The performance of a weightless neural network is tightly tied to the employed data preprocessing methods. So, different combinations of preprocessing techniques were tested to assess the WiSARD performance. Image preprocessing techniques: Sauvola binarization; adaptive Gaussian filter; Canny filter; and Otsu binarization. Audio preprocessing techniques: MFCC.

The main motivation for choosing weightless neural systems for this task was that they performed training and regression procedures faster than other learning models[195], which is highly desired for scenarios involving combinations of complex data, such as facial emotions, body language, audio, and text. The work here presented supports this motivation, given that Regression WiSARD could be trained with the whole preprocessed train set in less than 1h30 and generate the prediction of the whole test set in less than 40 minutes, once they have already been binarized. It can be inferred from the performance of ReW and CReW that through the proper exploitation of the preprocessing techniques, that is, the combination of their parameters, weightless neural systems can be considered a potential model for the fast and online resolution of the new and challenging questions that arise on the horizon of affective computing.

Results highlight that the solution using the only image was superior to the result of multi-modal architectures. This may have occurred due to the patterns generated by the audio preprocessing or due to noise created by the concatenation of image and audio signals in a single input. An architecture that processes modes or sub-patterns separately, similarly to NC-WiSARD, could alleviate this deficiency.

Other potential problem is the use of audio of actor in Personalized Track, where is supposed predict only the empathy of the listener, can imply in generating noise for the binary input, since the image-based approach performs better than the multi-modal approach in every scenario.

Currently, some researches are being made to overcome some issues that appeared during this task and others that aim to explore further the capability of weightless neural networks for empathy prediction. They are execution of prediction on the Generalized Track; usage of the transcription of conversations in the generation of the binary input; utilization of histogram descriptors (Local Binary Pattern, for example) in face preprocessing; testing of prediction policies in the CReW model involving the combination of results of ReW discriminators; exploration of the concatenation of multiple preprocessing methods of the same type (image or audio) into a single input; exploration of other combinations of hyperparameters in the preprocessing techniques already employed in this work; usage of other parameters in weightless neural systems (e.g., thresholds that limit the participation of a given memory location in prediction calculation depending on the number of zeros and ones in binary word); evaluation of the performance of ReW and CReW with other types of means (regression estimators) and kernels.

Chapter 7

Conclusion

Cognitive architectures aim to build agents whose features enable them to handle different tasks and environments autonomously. Such features include language, reasoning, problem-solving, decision-making, and planning. Such architectures are also fundamental to research related to machine consciousness. Naturally, the advancement of research in cognitive architectures depends on the available machine learning models.

Machine Learning has undergone major developments in recent years and achieved great performance in areas that until recently had no viable solutions. However, many of the main ML algorithms are extremely complex, especially its flagship, deep neural network, needing a lot of data to obtain a good performance, needing a considerably large training time[254], having great energy consumption[255], suffer from serious failures in security issues[256], in addition to the fact that we do not have a completely satisfactory explanation of how such algorithms work[32].

WiSARD is a type of neural network that responds to several of these challenges of deep nets, having online learning, ease of implementation in hardware, low memory cost, and with ease to implement forgetting mechanisms (as seen in Chapter 2). WiSARD is also a weightless artificial network, a type of ML model that has already proven itself efficient in building cognitive architectures[13, 16]. On the other hand, this model has some limitations due to the need for data binarization, the difficulty of dealing with data variance, and the fact that the model is just a classifier.

In this work, some contributions were made to expand the scope of use of WiSARD: its application in multi-label classification, the combination of WiSARD models in ensembles, WiSARD extensions for regression tasks, and its use in multi-modal prediction. The most significant contribution was the creation of an enhanced version of the n-tuple regression network adapted to use multiple kernel estimators and local *pdf* behaviors. Given the importance of machine learning regression for multiple tasks, such as statistical inference, feature importance, reinforcement learning, the WiSARD model needed to be extended to include regression in order to remain relevant and useful for consciousness research.

Some advantages and disadvantages of the WiSARD model to deep models that can be inferred from the experiments carried out in this thesis: a) advantages: fast training (disregarding preprocessing), easy implementation of forgetting mechanisms, easy handling for multi-modal data, storage of the whole training dataset implicitly, easy to deal with multi-output problems; b)disadvantages: difficulty in dealing with image variation and in learning intermediate representations of the data, total dependence on preprocessing techniques, choice of topology fully oriented to hyper-parameter exploration, impossibility of reusing models already trained in other domains.

In none of the experiments performed in this thesis WiSARD outperformed state-of-the-art in accuracy, despite being competitive in some datasets, however, in all of these tasks, WiSARD outperformed state-of-the-art in training time, even in some datasets training time has been high for WiSARD. However, given the high learning capability of the WiSARD model proven by its VC-dimension, this model is potentially competitive with the state-of-the-art. To reach this potential, further studies are needed regarding the pre-processing techniques used, as well as a greater understanding of the exact role of tuple selection in the model's topology.

All experiments reported in this thesis are reproducible, since all experiment code can be found at https://github.com/Lusquino/PhD-Thesis.git. The implementation of all models used in this thesis are map-based. The libraries used in these experiments are described in Appendix G.

7.1 Summary of the Thesis

Here is a summary of each chapter of the previous chapters:

7.1.1 Chapter 1

This chapter introduces the reader to the motivation of this work, giving a cognitive architecture's background, demonstrating the need to develop machine learning models in addition to the model paradigms. The classifier WiSARD was presented and located concerning the most used models in current machine learning scenario, especially deep neural network.

7.1.2 Chapter 2

A presentation of Weightless Artificial Neural Networks with some historical background. This chapter presents n-tuple classifier, the first model in this family. WiSARD's training and classification mechanisms were discussed in details, as well as their variations aimed at mitigating overfitting. There was also a survey on WiSARD extensions, such as ClusWiSARD for semi-supervised learning, clustering, and subprofile learning. Other WANNs have been explained. The chapter provides a recap on recent uses of WANNs.

Finally, this chapter discusses about the preprocessing techniques. Traditional WiSARD networks operate on base 2, therefore being able to handle only binary data, so that preprocessing is necessary both in the training phase and in the test phase. Although WiSARD extensions allow the increase of its base, the restriction on the mandatory preprocessing remains. The choice of binarization has a direct impact on an efficient representation of the data and, consequently, on the learning capacity and classification of the network. Given the importance of preprocessing for the WiSARD model, this chapter is specifically designed to describe the main binary strategies used in the experiments reported in this thesis.

7.1.3 Chapter 3

WiSARD and its ClusWiSARD extension were used in two multi-label classification architectures (Binary Relevance and Label Powerset) in a task of classifying facial muscles involved in expressing emotions. This challenge presents in particular the difficulty of dealing with some of the combinations of the original classes that distort their characteristics.

Label Powerset treats each combination of classes present in the training set as a class itself. At WiSARD this is done by creating new discriminators for each new combination of classes found. Binary Relevance deals with the presence of each class in the sample individually. WiSARD uses this approach and I try a network for each class with a presence discriminator and an absence discriminator. The advantages and disadvantages of these approaches are covered in this chapter.

Both weightless models were compared with state-of-the-art, showing competitive results in many classes, although the performance of WiSARD was poor in classes underrepresented in the dataset. In 10-fold cross validation, the best values found for accuracy and F1-score are, respectively, 89.66% and 49.11%.

7.1.4 Chapter 4

WiSARD and ClusWiSARD have been combined into five different types of ensemble (Bagging, Boosting, Borda Count, Tie-break, and Weighted Votes), all inspired by the traditional ensemble learning algorithms. Borda Count, Tie-break, and Weighted Votes ensembles have voting policies based on known voting systems. All of these ensembles and individual models have been validated in six datasets from widely varying domains, with the ensembles outperforming both original models in five of these. This was an initial exploration of WiSARD ensembles and no pruning technique was used, as the choice of parameters for the models used was random.

7.1.5 Chapter 5

Two non-parametric regression models based on WiSARD and n-tuple regression network were also created (Regression WiSARD and ClusRegression WiSARD), and they and their ensembles were validated in four datasets and compared with the famous GradientBoost and XGBoost techniques. The new models had very competitive results, with considerably less training time. In a palm oil prediction dataset, the WiSARD-based solution was outperformed by XGBoost in 0.011 in MAE, while outperforms it by four orders of magnitude at training time and two orders of magnitude at test time.

7.1.6 Chapter 6

Finally, Regression WiSARD was applied in a multi-modal emotion prediction challenge where it should predict the degree of empathy between interlocutors in each frame of a video. This is a very complex task, which involves a lot of background noise. There are few results in the literature involving this task, but the solution presented here has been compared with all of them and has shown promise. Weightless models were explored using only the image and the combination of image and audio. In the test set ReW achieved 0.08 of CCC, being competitive with the state-of-the-art, which obtained CCC 0.17 with a much more complex model.

7.2 WiSEMAN: Towards a WiSARD-based Cognitive Architecture

From the WiSARD's extensions achieved in this thesis, it was possible to envision a WiSARD-based cognitive architecture. This is called WiSEMAN (WiSARD Emotional Multi-Agent Network) and it is based on Aleksander's Kernel Architecture[12], which aims to create an agent with behavior similar to that of a conscious being, incorporating in all modules the consciousness requirements of the Fundamental Postulate of Consciousness[8]. The schema of this architecture is displayed in Lusquino et al.E.1

Each WiSEMAN agent has four operational modules:

- **Depiction:** Analogous to the process of empirical recognition of the environment, this module is responsible for interpreting the environment where the agent is. The multi-label and multi-modal systems developed in this thesis can be incorporated into this module;

- **Imagination:** This module proposes possible actions to be taken by the agent,considering its current state and the environment;
- **Emotion:** A quantitative evaluation of the action possibilities generated by the Imagination Module is made here, using a Regression WiSARD-based multidimensional RAM;
- **Attention:** In this module, each feature that forms the representation of the actions that can be performed from the states contemplated by the Imagination Module is evaluated separately.

Considering the use of WiSARD in an architecture that tries to satisfy the Fundamental Postulate and its corollaries, Regression WiSARD directly implies the possibility of satisfying the predictive capacity of the model. The multi-dimensional RAM neuron can also have learning fields and use a feedback function to meet the Will, Instinct, and Emotion corollaries, although such a solution is neurosymbolic and not strictly connectionist, as intended by Aleksander. WiSEMAN is more elaborated in Appendix E.

7.3 Final Remarks

In an attempt to expand the scope of the use of WiSARD this thesis made experiments both with systems that use the model and, therefore, externally expand its capacities (ensembles, multi-label and multi-modal), as well as with the model structure itself (non-parametric regression and logistic regression).

Both types of experimentation proved to be promising: (i) the use of WiSARD as an atomic component of a larger system, the model being capable of handling a variety of types of inputs, and using the score of its discriminators as function variables to generate more complex outputs; and (ii) the structure of WiSARD can still receive many extensions, with the addition of new content in its memory locations and functions to be applied on such content both in training and in classification. ReW introduced the possibility of using multiple contents per memory location and the combination of such contents.

The theoretical foundations behind the topology of the WiSARD models are not fully understood yet, but previous results show that it has a high learning potential[31, 116, 119]; a more comprehensive study of the choice of tuples can considerably expand the classification power of WiSARD. With a developed theoretical base, it would be possible to even expand the use of WiSARD as a training accelerator, embeddings generator, or component of hybrid systems with mainstream models, as it has already been used in other works[62, 63].

A more formal and extensive knowledge about mapping can expand the interpretability potential of the model (still limited to mental images), make it more suitable for neurosymbolic systems and partially solve the credit assignment problem, since each RAM can be understood as a feature, if there is some kind of semantic criterion in choosing the tuple.

It can be seen that despite the gains from the proposed extensions, WiSARD has some fundamental limitations, such as its dependence on preprocessing, which always implies the loss of information from the original data. In more complex datasets, hardly a single preprocessing technique will be enough to capture the relationship of the features and there is no guarantee that some combination of preprocessing can be efficient, since there is a lack of knowledge about the relationship between learning and the formation of n-tuples, in addition to this potentially introducing noise in the binary data and increasing the training and testing time, in addition to the extra preprocessing time, thus reducing the advantages of using WiSARD-like models. We saw this in Chapter 6.3.5, in which the use of only the image had a better result than the multi-modal solution. However, the concatenation of the binarization of the RGB channels in the tests with Cifar-10 shown in Chapter 6 has improved accuracy. The construction of a network that uses several WiSARD nodes that deal with different binarizations of the input, such as the NC-WiSARD[158], can be an adequate solution to this problem, especially in cases where the network needs robustness against rotation variance, translation, scale, etc.

Another significant limitation lies in bleaching, since the higher the training dataset and, supposedly, the more data for learning, the greater the chance that there will be a tie for a greater number of rounds, thus considerably increasing the classification time, or in the case of some variations of ClusWiSARD the training time as well. Thus, the creation of other tie-break mechanisms that circumvent this limitation can mitigate the current limitations of WiSARD, while increasing its main qualities.

Some branches of Roy Bhaskar's critical realism[257] have shown that ontology should precede epistemology in the scientific method. Its application in computer studies[258–261] has generated interesting observations about the need to study each computational model from its idiosyncrasies, that is, the analysis of a model must be done *a posteriori* and not from characteristics that are expected *a priori* that it has due to biases created in the community by the mainstream model.

The formation of a solid theory about n-tuple subspace must start from the model's characteristics and must not be contaminated by any prejudice from the community that originated from the deep model structure. This will allow the expansion of knowledge and the application of these models, as well as a mitigation of their natural limitations.

In addition to the WiSEMAN agent, the future works mentioned at the end of each chapter and a theoretical study on the features obtained through the mapping, mentioned in the previous section, there is another interesting derivation of this thesis, which is an ongoing work. An important aspect of the creation of regression models was the expansion of the neuron RAMs of the traditional WiSARD to multi-dimensional RAMs, inspired by the Kolcz model[145]. This neuron can store different types of information and can use any function to generate a continuous output. This new type of neuron can be used in a WiSARD-based multi-layer architecture (it is possible to use any loss function in place of the kernel estimator used for non-parametric regression), in reinforcement learning tasks, in transfer learning and, may even have storage fields linked to emotional feedback or any other type of specialized contempt based on specific domains.

References

[1] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Papers of the Eastern Joint IRE-AIEE-ACM Computer Conference*, pp. 225–232, 1959.

[2] L. A. Lusquino Filho, L. F. Oliveira, A. Lima Filho, G. P. Guarisa, L. M. Felix, P. M. Lima, and F. M. França, "Extending the weightless wisard classifier for regression," *Neurocomputing*, vol. 416, pp. 280–291, 2020.

[3] M. Valstar and M. Pantic, "Induced disgust, happiness and surprise: an addition to the mmi facial expression database," in *Proc. 3rd Intern. Workshop on EMOTION (satellite of LREC)*, p. 65, Paris, France., 2010.

[4] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proc. IEEE Intl Conference Face and Gesture Recognition (AFGR 00)*, pp. 46–53, 2000.

[5] P. Barros, N. Churamani, E. Lakomkin, H. Siqueira, A. Sutherland, and S. Wermter, "The omg-emotion behavior dataset," in *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN 2018)*, pp. 1408–1414, 2018.

[6] I. Aleksander, W. Thomas, and P. Bowden, "Wisard, a radical new step forward in image recognition," *Sensor Rev.*, vol. 4(3), pp. 120–124, 1984.

[7] I. Aleksander, M. D. Gregorio, F. França, P. Lima, and H. Morton, "A brief introduction to weightless neural systems," in *Proceedings of the 17th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 299–305, 2009.

[8] I. Aleksander, "Artificial consciousness: An update," in *Proc. of IWANN*, vol. 3, pp. 566–583, 1995.

[9] I. Aleksander and D. Gamez, "Iconic training and effective information: Evaluating meaning in discrete neural networks," in *2009 AAAI Fall Symposium Series*, 2009.

[10] D. Gamez and I. Aleksander, “Accuracy and performance of the state-based ϕ and liveliness measures of information integration,” *Consciousness and Cognition*, vol. 20, no. 4, pp. 1403–1424, 2011.

[11] I. Aleksander and D. Gamez, “Informational theories of consciousness: a review and extension,” in *From brains to systems*, pp. 139–147, Springer, 2011.

[12] I. Aleksander and H. Morton, “Axiomatic consciousness theory for visual phenomenology in artificial intelligence,” in *AAAI Fall Symposium: AI and Consciousness*, pp. 18–23, 2007.

[13] I. Aleksander and H. Morton, “Phenomenal weightless machines.,” in *ESANN*, Citeseer, 2009.

[14] D. Gamez and I. Aleksander, “Taking a mental stance towards artificial systems,” in *2009 AAAI Fall Symposium Series*, 2009.

[15] I. Aleksander and D. Gamez, “Iconic training and effective information: Evaluating meaning in discrete neural networks,” in *2009 AAAI Fall Symposium Series*, 2009.

[16] M. Shanahan, “Consciousness, emotion, and imagination: a brain-inspired architecture for cognitive robotics,” in *In Proceedings of the AISB’05 Workshop: Next Generation Approaches to Machine Consciousness*, Citeseer, 2005.

[17] G. F. Ellis, “Top-down causation and the human brain,” in *Downward causation and the neurobiology of free will*, pp. 63–81, Springer, 2009.

[18] K. Rauss and G. Pourtois, “What is bottom-up and what is top-down in predictive coding?,” *Frontiers in psychology*, vol. 4, p. 276, 2013.

[19] K. Rauss and G. Pourtois, “What is bottom-up and what is top-down in predictive coding?,” *Frontiers in psychology*, vol. 4, p. 276, 2013.

[20] T. Sikkens, C. A. Bosman, and U. Olcese, “The role of top-down modulation in shaping sensory processing across brain states: implications for consciousness,” *Frontiers in systems neuroscience*, vol. 13, p. 31, 2019.

[21] I. Panella, L. Z. Fragonara, and A. Tsourdos, “A deep learning cognitive architecture: Towards a unified theory of cognition,” in *Proceedings of SAI Intelligent Systems Conference*, pp. 566–582, Springer, 2020.

[22] A. Newell, “Précis of unified theories of cognition,” *Behavioral and Brain Sciences*, vol. 15, no. 3, pp. 425–437, 1992.

[23] A. Newell, "Unified theories of cognition and the role of soar," in *SOAR: A cognitive architecture in perspective*, pp. 25–79, Springer, 1992.

[24] A. Newell, *Unified theories of cognition.* Harvard University Press, 1994.

[25] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE transactions on evolutionary computation*, vol. 11, no. 2, pp. 151–180, 2007.

[26] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.

[27] A. Lieto, M. Bhatt, A. Oltramari, and D. Vernon, "The role of cognitive architectures in general artificial intelligence," 2018.

[28] P. Ye, T. Wang, and F.-Y. Wang, "A survey of cognitive architectures in the past 20 years," *IEEE transactions on cybernetics*, vol. 48, no. 12, pp. 3280–3290, 2018.

[29] R. Kingdon, "A review of cognitive architectures," *ISO Project report*, 2008.

[30] G. A. Kelly, *The psychology of personal constructs. Volume 1: A theory of personality.* 500 Fifth Avenue, New York: WW Norton and Company, 1955.

[31] H. C. C. Carneiro, C. E. Pedreira, F. M. G. França, and P. M. V. Lima, "The exact vc dimension of the wisard n-tuple classifier," in *Neural Computation*, vol. 31, Issue 1, pp. 176–207, 2019.

[32] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[35] S. J. Pan, J. T. Kwok, Q. Yang, *et al.*, "Transfer learning via dimensionality reduction.," in *AAAI*, vol. 8, pp. 677–682, 2008.

[36] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[37] M. Oza, H. Vaghela, and K. Srivastava, "Progressive generative adversarial binary networks for music generation," in *International Conference on Innovative Computing and Communications*, pp. 181–192, Springer, 2020.

[38] Y. Wang, P. Bilinski, F. Bremond, and A. Dantcheva, "Imaginator: Conditional spatio-temporal gan for video generation," in *The IEEE Winter Conference on Applications of Computer Vision*, pp. 1160–1169, 2020.

[39] M. A. Haidar and M. Rezagholizadeh, "Textkd-gan: Text generation using knowledge distillation and generative adversarial networks," in *Canadian Conference on Artificial Intelligence*, pp. 107–118, Springer, 2019.

[40] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, pp. 3856–3866, 2017.

[41] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*, pp. 273–278, IEEE, 2013.

[42] A. Bérard, O. Pietquin, C. Servan, and L. Besacier, "Listen and translate: A proof of concept for end-to-end speech-to-text translation," *arXiv preprint arXiv:1612.01744*, 2016.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[44] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[45] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[46] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, *et al.*, "A generative vision model that trains with high data efficiency and breaks text-based captchas," *Science*, vol. 358, no. 6368, 2017.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[48] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," *arXiv preprint arXiv:1706.04317*, 2017.

[49] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[50] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328*, 2017.

[51] B. Lake and M. Baroni, "Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks," 2018.

[52] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[53] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[54] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[55] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[56] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[57] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[58] B. Zohuri and F. M. Rahmani, "Artificial intelligence driven resiliency with machine learning and deep learning components," *International Journal of Nanotechnology & Nanomedicine*, vol. 4, no. 2, pp. 1–8, 2019.

[59] F. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *arXiv preprint arXiv:2001.02522*, 2020.

[60] W. Zheng, H. Liu, B. Wang, and F. Sun, "Cross-modal learning for material perception using deep extreme learning machine," *International Journal of Machine Learning and Cybernetics*, pp. 1–11, 2019.

[61] C. Zhang, Q. Dai, and G. Song, "Deepcascade-wr: a cascading deep architecture based on weak results for time series prediction," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 825–840, 2020.

[62] A. T. L. Bacellar, B. F. Goldstein, V. C. Ferreira, L. Santiago, P. Lima, and F. França, "Fast deep neural networks convergence using a weightless neural model," in *ESANN*, 2020.

[63] S. Milhomem, T. d. S. Almeida, W. G. da Silva, E. M. da Silva, and R. L. de Carvalho, "Weightless neural network with transfer learning to detect distress in asphalt," *arXiv preprint arXiv:1901.03660*, 2019.

[64] A. Sivasankari, S. Sudarvizhi, and S. R. A. Bai, "Comparative study of different clustering and decision tree for data mining algorithm," *International Journal of Computer Science and Information Technology Research*, vol. 2, no. 3, pp. 221–232, 2014.

[65] A. Pradhan, "Support vector machine - a survey," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 8, pp. 82–85, 2012.

[66] C. Bielza and P. Larrañaga, "Discrete bayesian network classifiers: a survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–43, 2014.

[67] D. L. Abd AL-Nabi and S. S. Ahmed, "Survey on classification algorithms for data mining:(comparison and evaluation)," *International Journal of Computer Engineering and Intelligent Systems*, vol. 4, no. 8, pp. 18–27, 2013.

[68] L. S. de Araújo, V. C. Patil, C. B. Prado, T. A. Alves, L. A. Marzulo, F. M. França, and S. Kundu, "Design of robust, high-entropy strong pufs via weightless neural network," *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 235–249, 2019.

[69] L. Santiago, V. C. Patil, C. B. Prado, T. A. Alves, L. A. Marzulo, F. M. França, and S. Kundu, "Realizing strong puf from weak puf via neural computing," in *2017 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT)*, pp. 1–6, IEEE, 2017.

[70] V. C. Ferreira, A. S. Nery, L. A. J. Marzulo, L. Santiago, D. Souza, B. F. Goldstein, F. M. G. França, and V. Alves, "A feasible fpga weightless neural accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2019.

[71] X. Chen, D. Z. Chen, and X. S. Hu, "modnn: Memory optimal dnn training on gpus," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 13–18, 2018.

[72] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[73] M. G. F. Coutinho, M. F. Torquato, and M. A. C. Fernandes, "Deep neural network hardware implementation based on stacked sparse autoencoder," *IEEE Access*, vol. 7, pp. 40674–40694, 2019.

[74] F. Yang, Z. Zhang, H. Wang, Y. Li, and X. Hu, "Xdeep: An interpretation tool for deep neural networks," *arXiv preprint arXiv:1911.01005*, 2019.

[75] V. G. T. da Costa, A. C. P. de Leon Ferreira, S. B. Junior, *et al.*, "Strict very fast decision tree: a memory conservative algorithm for data stream mining," *Pattern Recognition Letters*, vol. 116, pp. 22–28, 2018.

[76] O. Asian, O. T. Yildiz, and E. Alpaydin, "Calculating the vc-dimension of decision trees," in *2009 24th International Symposium on Computer and Information Sciences*, pp. 193–198, IEEE, 2009.

[77] J. Struharik, "Implementing decision trees in hardware," in *2011 IEEE 9th International Symposium on Intelligent Systems and Informatics*, pp. 41–46, IEEE, 2011.

[78] M. Moshkovitz, Y.-Y. Yang, and K. Chaudhuri, "Connecting interpretability and robustness in decision trees through separation," 2021.

[79] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, 1992.

[80] N. Bajaj, G. T.-C. Chiu, and J. P. Allebach, "Reduction of memory footprint and computation time for embedded support vector machine (svm) by kernel expansion and consolidation," in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2014.

[81] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[82] S. Kumar, J. Manikandan, and V. K. Agrawal, "Hardware implementation of support vector machine classifier using reconfigurable architecture," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 45–50, 2017.

[83] D.-H. Nguyen and M.-T. Le, "Improving the interpretability of support vector machines-based fuzzy rules," *arXiv preprint arXiv:1408.5246*, 2014.

[84] D. Roth, "Learning in natural language," in *IJCAI*, pp. 898–904, 1999.

[85] H. R. Seth and H. Banka, "Hardware implementation of naïve bayes classifier: A cost effective technique," in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pp. 264–267, 2016.

[86] C. Molnar, *Interpretable Machine Learning.* https://christophm.github.io/interpretable-ml-book/, 2018. https://christophm.github.io/interpretable-ml-book/.

[87] L. I. Kuncheva and M. Galar, "Theoretical and empirical criteria for the edited nearest neighbour classifier," in *2015 IEEE International Conference on Data Mining*, pp. 817–822, IEEE, 2015.

[88] J. Saikia, S. Yin, Z. Jiang, M. Seok, and J.-s. Seo, "K-nearest neighbor hardware accelerator using in-memory computing sram," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2019.

[89] T. S. Kuhn, "The structure of scientific revolutions. 50th anniversary," *Argument: Biannual Philosophical Journal*, vol. 3, no. 2, pp. 539–543, 2013.

[90] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[91] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[92] J. Hadamard, *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*, vol. 33. Paris, Avenue du President Kennedy: Imprimerie nationale, 1908.

[93] H. J. Kelley, "Gradient theory of optimal flight paths," *Ars Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[94] A. E. Bryson, "A gradient method for optimizing multi-stage allocation processes," in *Proc. Harvard Univ. Symposium on digital computers and their applications*, vol. 72, p. 22, 1961.

[95] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamrelidze, and E. F. Mishchenko, *Mathematical theory of optimal processes*. Abingdon, England; New York: Routledge, 1961.

[96] A. E. Bryson and W. F. Denham, "A steepest-ascent method for solving optimum programming problems," 1962.

[97] S. Dreyfus, "The numerical solution of variational problems," *Journal of Mathematical Analysis and Applications*, vol. 5, no. 1, pp. 30–45, 1962.

[98] J. H. Wilkinson, *The algebraic eigenvalue problem*, vol. 87. Oxford: Clarendon Press Oxford, 1965.

[99] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Transactions on Electronic Computers*, no. 3, pp. 299–307, 1967.

[100] S. Director and R. Rohrer, "Automated network design-the frequency-domain case," *IEEE Transactions on Circuit Theory*, vol. 16, no. 3, pp. 330–337, 1969.

[101] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. Boca Raton, Florida: CRC Press, 1975.

[102] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors," *Master's Thesis (in Finnish), Univ. Helsinki*, pp. 6–7, 1970.

[103] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[104] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron," *IEICE Technical Report, A*, vol. 62, no. 10, pp. 658–665, 1979.

[105] K. Fukushima, "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980.

[106] R. M. Haralick and A. C. Yuksel, "The n-tuple subspace classifier: Extensions and survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[107] L. A. D. Lusquino Filho, "Classificação de emoções faciais utilizando a rede neural sem pesos wisard," Master's thesis, Universidade Federal do Rio de Janeiro, 2018.

[108] W. Bledsoe and C. Bisson, "Improved memory matrices for the n-tuple pattern recognition method," *IRE Transactions on Electronic Computers*, no. 3, pp. 414–415, 1962.

[109] B. Grieco, P. Lima, M. De Gregorio, and F. França, "Extracting fuzzy rules from "mental" images generated by modified wisard perceptrons," in *Proc. of ESANN*, vol. 16, pp. 101–773, 2008.

[110] D. F. P. de Souza, H. C. C. Carneiro, F. M. G. França, and P. M. V. Lima, "Rock-paper-scissors wisard," in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI 2013 & CBIC 2013)*, pp. 178–182, 2013.

[111] J. Bishop, A. Crowe, P. Michinton, and R. Mitchell, "Evolutionary learning to optimise mapping in n-tuple networks," in *IEE Colloquium on Machine Learning*, pp. 3–1, IET, 1990.

[112] M. Giordano and M. De Gregorio, "An evolutionary approach for optimizing weightless neural networks.," in *ESANN*, 2019.

[113] C. Soares, C. da Silva, M. De Gregorio, and F. França, "Uma implementação em software do classificador wisard," *V Simpósio Brasileiro de Redes Neurais*, vol. 2, pp. 225–229, 1998.

[114] H. B. Azhar and K. Dimond, "A stochastic search algorithm to optimize an n-tuple classifier by selecting its inputs," in *International Conference Image Analysis and Recognition*, pp. 556–563, Springer, 2004.

[115] M. H. B. Azhar, F. Deravi, and K. Dimond, "Criticality dispersion in swarms to optimize n-tuples," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 1–8, 2008.

[116] G. P. Guarisa, “Estudo comparativo de técnicas de mapeamento no classificador wisard,” Master’s thesis, Universidade Federal do Rio de Janeiro, 2020.

[117] M. Ettaouil and C. Loqman, “A new optimization model for solving the constraint satisfaction problem,” *Journal of Advanced Research in Computer Science*, vol. 1, no. 1, pp. 13–31, 2009.

[118] V. N. Vapnik and A. Y. Chervonenkis, “On the uniform convergence of relative frequencies of events to their probabilities,” in *Measures of complexity*, pp. 11–30, Springer, 2015.

[119] N. P. Bradshaw, *An Analysis in Weightless Neural Systems.* PhD thesis, Imperial College London, 1996.

[120] G. A. Carpenter, S. Grossberg, and D. B. Rosen, “Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system,” *Neural networks*, vol. 4, no. 6, pp. 759–771, 1991.

[121] I. Wickert and F. M. França, “Autowisard: Unsupervised modes for the wisard,” in *International Work-Conference on Artificial Neural Networks*, pp. 435–441, Springer, 2001.

[122] I. Wickert and F. M. França, “Validating an unsupervised weightless perceptron,” in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP’02.*, vol. 2, pp. 537–541, IEEE, 2002.

[123] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[124] D. O. Cardoso, D. Carvalho, D. S. F. Alves, D. F. P. de Souza, H. C. C. Carneiro, C. E. Pedreira, P. M. V. Lima, and F. M. G. França, “Financial credit analysis via a clustering weightless neural classifier,” *Neurocomputing*, vol. 183, pp. 70–78, 2016.

[125] L. A. D. Lusquino Filho, G. P. Guarisa, A. Lima Filho, L. F. R. de Oliveira, F. M. G. França, and P. M. V. Lima, “Classifying actions units with cluswisard,” in *Proceedings of the 28th International Conference on Artificial Neural Networks*, 2019.

[126] D. d. O. Cardoso, P. M. Lima, M. De Gregorio, J. Gama, and F. M. França, “Clustering data streams with weightless neural networks,” in *ESANN*

2011, 19th European Symposium on Artificial Neural Networks, (Bruges, Belgium), pp. 201 – 206, 2011.

[127] D. Cardoso, M. De Gregorio, P. Lima, J. Gama, and F. França, "A weightless neural network-based approach for stream data clustering," in *Intelligent Data Engineering and Automated Learning - IDEAL 2012 - 13th International Conference*, (Natal, Brazil), pp. 328—-335, 2012.

[128] D. O. Cardoso, F. M. França, and J. Gama, "Wcds: A two-phase weightless neural system for data stream clustering," *New Generation Computing*, vol. 35, no. 4, pp. 391–416, 2017.

[129] D. O. Cardoso, J. Gama, and F. M. França, "Weightless neural networks for open set recognition," *Machine Learning*, vol. 106, no. 9-10, pp. 1547–1567, 2017.

[130] D. F. P. de Souza, F. M. G. França, and P. M. V. Lima, "Spatio-temporal pattern classification with kernelcanvas and wisard," in *2014 Brazilian Conference on Intelligent Systems (BRACIS 2014)*, pp. 228–233, 2014.

[131] L. A. Lusquino Filho, L. F. Oliveira, H. C. Carneiro, G. P. Guarisa, A. Lima Filho, F. M. França, and P. M. Lima, "A weightless regression system for predicting multi-modal empathy," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)(FG)*, pp. 554–558, 2020.

[132] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[133] D. F. P. de Souza, F. M. G. França, and P. M. V. Lima, "Real-time music tracking based on a weightless neural network," in *Proceedings of the 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, 2015.

[134] A. Kolcz and A. N.M., "n-tuple regression network," *Neural Networks*, vol. 9, pp. 855–869, 1996.

[135] B. Widrow and S. Stearns, *Adaptive signal processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[136] W. Härdle, *Applied nonparametric regression.* No. 19, University Printing House Shaftesbury Road Cambridge CB2 8BS United Kingdom: Cambridge University Press, 1990.

[137] R. A. Davis, K.-S. Lii, and D. N. Politis, "Remarks on some nonparametric estimates of a density function," in *Selected Works of Murray Rosenblatt*, pp. 95–100, Springer, 2011.

[138] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[139] T. Cacoullos, "Estimation of a multivariate density," tech. rep., University of Minnesota, 1964.

[140] D. J. Hand, "Kernel discriminant analysis.," *John Wiley & Sons, Inc., One Wiley Dr., Somerset, N. J. 08873, 1982, 264*, 1982.

[141] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization.* Inc. 90 Eglinton Ave.: John Wiley & Sons, 2015.

[142] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[143] G. S. Watson, "Smooth regression analysis," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[144] D. F. Specht *et al.*, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[145] A. Kolcz, *Approximation Properties of Memory-based.* PhD thesis, University of Manchester - Institute of Science and Technology, Manchester, United Kingdom, 1996.

[146] A. Kolcz and N. Allinson, "Distance relationships in the n-tuple mapping," *Pattern Recognition*, 1995.

[147] N. Allinson, M. J. Johnson, *et al.*, "Self-organising n-tuple feature maps," *Neural Network World*, vol. 5, pp. 511–530, 1993.

[148] G. D. Tattersall, S. Foster, and R. D. Johnston, "Single-layer lookup perceptrons," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 138, pp. 46–54, IET, 1991.

[149] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[150] L. Santiago, L. Verona, F. Rangel, F. Firmino, D. S. Menasche, W. Caarls, M. B. Jr, S. Kundu, P. M. Lima, and F. M. França, "Memory efficient

weightless neural network using bloom filter," in *27 th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2019.

[151] L. Santiago, L. Verona, F. Rangel, F. Firmino, D. S. Menasché, W. Caarls, M. Breternitz Jr, S. Kundu, P. M. Lima, and F. M. França, "Weightless neural networks as memory segmented bloom filters," *Neurocomputing*, 2020.

[152] U. Manber and S. Wu, "An algorithm for approximate membership checking with application to password security," *Information Processing Letters*, vol. 50, no. 4, pp. 191–197, 1994.

[153] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[154] P. C. Dillinger and P. Manolios, "Bloom filters in probabilistic verification," in *International Conference on Formal Methods in Computer-Aided Design*, pp. 367–381, Springer, 2004.

[155] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.

[156] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pp. 75–88, 2014.

[157] A. Geil, M. Farach-Colton, and J. D. Owens, "Quotient filters: Approximate membership queries on the gpu," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 451–462, IEEE, 2018.

[158] L. C. Bandeira, H. L. França, F. M. França, and C. Computaçao, "Nc-wisard: Uma interpretação booleana da arquitetura neocognitron," in *Anais do IX Congresso Brasileiro de Redes Neurais/Inteligência Computacional.*

[159] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[160] I. Aleksander and W. Kan, "A probabilistic logic neuron network for associative learning," in *Proceedings of the First Int. Conf. on Neural Networks*, pp. 541–548, IEEE, 1987.

[161] I. Aleksander, *An introduction to neural computing.* London: Chapman and Hall, 1990.

[162] J. G. Taylor, “Spontaneous behaviour in neural networks,” *Journal of Theoretical Biology*, vol. 36, no. 3, pp. 513–528, 1972.

[163] F. ECDBC, D. Bisset, and M. Fairhurst, “A goal seeking neuron for boolean neural networks,” in *International Neural Network Conference*, pp. 894–897, Springer, 1990.

[164] I. Aleksander, “Ideal neurons for neural computers,” *Parallel Processing in Neural Systems and Computers*, pp. 225–228, 1990.

[165] J. Mrsic-Flogel, “Approaching cognitive system design,” in *Proceedings of the International Conference on Artificial Neural Networks (ICANN 91)*, pp. 879–883, 1991.

[166] I. Aleksander and H. Morton, “General neural unit: retrieval performance,” *Electronics letters*, vol. 27, no. 19, pp. 1776–1778, 1991.

[167] I. Aleksander, *The world in my mind, my mind in the world.* Mall Luton: Andrews UK Limited, 2013.

[168] J. Austin, “Adam: A distributed associative memory tor scene analysis,” in *pp. IV-285 in Proceedings of First International Conference on Neural Networks, ed. M. Caudill, C. Butler, IEEE, San Diego (June, 1987)*, 1987.

[169] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, “Non-holographic associative memory,” *Nature*, vol. 222, no. 5197, pp. 960–962, 1969.

[170] A. Alarcón-Paredes and A.-J. Argüelles-Cruz, “Cainn-weightless alpha-beta neural network,” in *2008 Electronics, Robotics and Automotive Mechanics Conference (CERMA’08)*, pp. 434–438, IEEE, 2008.

[171] P. Kanerva, *Sparse distributed memory.* Rogers Street, Cambridge: MIT press, 1988.

[172] J. Tome, “Neural activation ratio based fuzzy reasoning,” in *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36228)*, vol. 2, pp. 1217–1222, IEEE, 1998.

[173] W. R. de Oliveira, “Quantum ram based neural netoworks.,” in *ESANN*, vol. 9, pp. 331–336, 2009.

[174] D. N. Nascimento, R. L. de Carvalho, Mora-Camino, P. M. V. F., Lima, and F. M. G. França, “A wisard-based multi-term memory framework for online tracking of objects,” in *Proceedings of the 23rd European Symposium*

on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 19–24, 2015.

[175] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[176] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in neural information processing systems*, pp. 547–553, 2000.

[177] R. Barbosa, D. O. Cardoso, D. Carvalho, and F. M. França, "A neuro-symbolic approach to gps trajectory classification," ESANN, 2017.

[178] R. Barbosa, D. O. Cardoso, D. Carvalho, and F. M. França, "Weightless neuro-symbolic gps trajectory classification," *Neurocomputing*, vol. 298, pp. 100–108, 2018.

[179] H. C. Carneiro, F. M. França, and P. M. Lima, "Wann-tagger-a weightless artificial neural network tagger for the portuguese language," in *International Conference on Neural Computation*, vol. 2, pp. 330–335, SciTePress, 2010.

[180] H. C. C. Carneiro, F. M. G. França, and P. M. V. Lima, "Multilingual part-of-speech tagging with weightless neural networks," *Neural Networks*, vol. 66, pp. 11–21, 2015.

[181] H. C. C. Carneiro, C. E. Pedreira, F. M. G. França, and P. M. V. Lima, "A universal multilingual weightless neural network tagger via quantitative linguistics," *Neural Networks*, vol. 91, pp. 85–101, 2017.

[182] Y. Yusof, H. A. H. Mansor, and A. Ahmad, "Utilizing unsupervised weightless neural network as autonomous states classifier in reinforcement learning algorithm," in *2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA)*, pp. 264–269, IEEE, 2017.

[183] Y. Yusof, H. A. H. Mansor, and H. D. Baba, "Applying hybrid reinforcement and unsupervised wieghtless neural network learning algorithm on autonomous mobile robot navigation.," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 1-3, pp. 133–138, 2017.

[184] Y. Yusof, H. A. H. Mansor, and H. D. Baba, "Simulation of mobile robot navigation utilizing reinforcement and unsupervised weightless neural network learning algorithm," in *2015 IEEE Student Conference on Research and Development (SCOReD)*, pp. 123–128, IEEE, 2015.

[185] M. De Gregorio and M. Giordano, “Memory transfer in drasiw–like systems,” in *Proceedings ESANN*, p. 25, Presses universitaires de Louvain, 2015.

[186] J. Gryak, R. M. Haralick, and D. Kahrobaei, “Solving the conjugacy decision problem via machine learning,” *Experimental Mathematics*, vol. 29, no. 1, pp. 66–78, 2020.

[187] R. Cheruku, D. R. Edla, V. Kuppili, R. Dharavath, and N. R. Beechu, “Automatic disease diagnosis using optimised weightless neural networks for low-power wearable devices,” *Healthcare technology letters*, vol. 4, no. 4, pp. 122–128, 2017.

[188] N. G. Haider and A. Karim, “Ram based neural-network controlled vehicle: path-tracking & collision avoidance,” in *2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)*, pp. 1–8, IEEE, 2013.

[189] J.-C. Yen, F.-J. Chang, and S. Chang, “A new criterion for automatic multilevel thresholding,” *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 370–378, 1995.

[190] J. Sauvola and M. Pietikainen, “Adaptive document image binarization,” *Pattern Recognition*, vol. 33, 2000.

[191] H. P. Luhn, “A statistical approach to mechanized encoding and searching of literary information,” *IBM Journal of research and development*, vol. 1, no. 4, pp. 309–317, 1957.

[192] K. S. Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of documentation*, 1972.

[193] A. Kappaun, K. Camargo, F. Rangel, F. Firmino, P. M. V. Lima, and J. Oliveira, “Evaluating binary encoding techniques for wisard,” in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 103–108, IEEE, 2016.

[194] L. A. D. Lusquino Filho, F. M. G. França, and P. M. V. Lima, “Near-optimal facial emotion classification using wisard-based weightless system,” in *Proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 85–90, 2018.

[195] L. A. D. Lusquino Filho, L. F. R. Oliveira, A. Lima Filho, G. P. Guarisa, P. M. V. Lima, , and F. M. G. França, “Prediction of palm oil production with an enhanced *n*-tuple regression network,” in *Proceedings of the*

27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 301–306, 2019.

[196] P. Ekman and F. W.V, "Manual for the facial action coding system," 1977.

[197] V. Bettadapura, "Face expression recognition and analysis: The state of the art."

[198] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*, pp. 667–685, Springer, 2009.

[199] K. R. Scherer, *Handbook of methods in nonverbal behavior research.* Cambridge University Press, 1985.

[200] Q. Li, J. Yu, T. Kurihara, and S. Zhan, "Micro-expression analysis by fusing deep convolutional neural network and optical flow," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 265–270, IEEE, 2018.

[201] X. Xu, C. Quan, and F. Ren, "Facial expression recognition based on gabor wavelet transform and histogram of oriented gradients," in *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 2117–2122, IEEE, 2015.

[202] L.-J. Xie, H. Wen, and N.-F. Xiao, "Affective computing model based on hmm for home-service robot," *Computer Engineering and Design*, vol. 33, no. 1, pp. 322–327, 2012.

[203] R. Asmara, P. Choirina, C. Rahmad, A. Setiawan, F. Rahutomo, R. Yusron, and U. Rosiani, "Study of drmf and asm facial landmark point for micro expression recognition using klt tracking point feature," in *Journal of Physics: Conference Series*, vol. 1402, p. 077039, IOP Publishing, 2019.

[204] H. Tao and T. S. Huang, "Connected vibrations: a modal analysis approach for non-rigid motion tracking," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pp. 735–740, IEEE, 1998.

[205] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2, pp. 131–163, 1997.

[206] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[207] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[208] P. Khorrami, T. L. Paine, and T. S. Huang, “Do deep neural networks learn facial action units when doing expression recognition?,” 2017.

[209] P. Prajod, D. Schiller, T. Huber, and E. André, “Do deep neural networks forget facial action units? – exploring the effects of transfer learning in health related facial expression recognition,” 2021.

[210] J.-Y. Chang and J.-L. Chen, “A facial expression recognition system using neural networks,” in *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, vol. 5, pp. 3511–3516 vol.5, 1999.

[211] B. Martinez, M. F. Valstar, B. Jiang, and M. Pantic, “Automatic analysis of facial actions: A survey,” in *IEEE Transactions on Affective Computing*, 2018.

[212] W.-S. Chu, F. D. la Torre, and J. F. Cohn, “Learning spatial and temporal cues for multi-label facial action unit detection,” 2016.

[213] T. Almaev, B. Martinez, and M. Valstar, “Learning to transfer: Transferring latent task structures and its application to person-specific facial action unit detection,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 3774–3782, 2015.

[214] R. Breuer and R. Kimmel, “A deep learning perspective on the origin of facial expressions,” 2017.

[215] I. Abbasnejad, S. Sridharan, D. Nguyen, S. Denman, C. Fookes, and S. Lucey, “Using synthetic data to improve facial expression analysis with 3d convolutional networks,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[216] G. Pons and D. Masip, “Multi-task, multi-label and multi-domain learning with residual convolutional networks for emotion recognition,” 2018.

[217] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[218] D. Opitz and R. Maclin, “Popular ensemble methods: An empirical study,” *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[219] R. da Silva Moreira and N. F. F. Ebecken, "Maritime vessel tracking with an ensemble of wisard classifiers in video," *International Journal of Systems Applications, Engineering and Development*, vol. 9, 2015.

[220] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(10), pp. 993–1001, 1990.

[221] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, 2018. Issue 4, e1249.

[222] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24(2), pp. 123–140, 1996.

[223] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5(2), pp. 197–227, 1990.

[224] L. Breiman, "Arcing classifier (with discussion and a rejoinder by the author)," *The annals of statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[225] R. Schapire, Y. Freund, *et al.*, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Second European Conference on Computational Learning Theory*, pp. 23–37, 1995.

[226] D. Lippman, "Voting theory," *Creative Commons BY-SA (A summary document)*, 2013.

[227] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[228] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[229] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[230] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[231] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[232] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” *arXiv preprint arXiv:2010.01412*, 2020.

[233] P. Burkert, F. Trier, M. Z. Afzal, A. Dengel, and M. Liwicki, “Dexpression: Deep convolutional neural network for expression recognition,” *arXiv preprint arXiv:1509.05371*, 2015.

[234] A. Byerly, T. Kalganova, and I. Dear, “No routing needed between capsules,” 2021.

[235] M. S. Tanveer, M. U. K. Khan, and C.-M. Kyung, “Fine-tuning darts for image classification,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4789–4796, IEEE, 2021.

[236] T. Thongtan and T. Phienthrakul, “Sentiment classification using document embeddings trained with cosine similarity,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pp. 407–414, 2019.

[237] “Kdd br competition 2018.” https://www.kaggle.com/c/kddbr-2018/.

[238] M. C. Mackey and L. Glass, “Oscillation and chaos in physiological control systems,” *Science*, vol. 197, pp. 287–289, 1977.

[239] E. Fix and J. L. Hodges, “Discriminatory analysis. nonparametric discrimination: Consistency properties,” *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[240] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16*, pp. 785–794, 2016.

[241] M. Awad and R. Khanna, “Support vector regression,” in *Efficient learning machines*, pp. 67–80, Springer, 2015.

[242] “House prices: Advanced regression techniques.”

[243] “Calcofi: Over 60 years of oceanographic data.”

[244] “Parkinson’s telemonitoring dataset.”

[245] L. Breiman, "Arcing the edge," in *Technical Report 486, Statistics Department, University of California, Berkeley*, 1997.

[246] J. Friedman, "Greedy function approximation: A gradient boosting machine," in *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001.

[247] "Kaggle display advertising challenge dataset." http://labs.criteo.com/2014/02/kaggle-displayadvertising-challenge-dataset/.

[248] A. Lima Filho, G. P. Guarisa, L. A. D. Lusquino Filho, L. F. R. Oliveira, C. A. N. Cosenza, F. M. G. França, and P. M. V. Lima, "Interpretation of model agnostic classifiers via local mental images," in *Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020.

[249] R. N. Rocha, L. Leopoldo Filho, M. Aredes, F. M. França, and P. M. Lima, "Regression wisard application of controller on dc statcom converter under fault conditions," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 860–867, IEEE, 2020.

[250] D. Dresvyanskiy, E. Ryumina, H. Kaya, M. Markitantov, A. Karpov, and W. Minker, "An audio-video deep and transfer learning framework for multimodal emotion recognition in the wild," 2020.

[251] Z. Yu and N. Shi, "A multi-modal deep learning model for video thumbnail selection," 2020.

[252] S. Moon, S. Kim, and H. Wang, "Multimodal transfer deep learning for au-dio visual recognition," *arXiv preprint arXiv:1412.3121*, 2014.

[253] P. Barros, N. Churamani, E. Lakomkin, H. Siqueira, A. Sutherland, and S. Wermter, *Results of the 2018 OMG-Empathy Prediction Challenge*, 2019 (accessed September 27, 2020).

[254] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[255] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[256] Y. He, G. Meng, K. Chen, X. Hu, and J. He, "Towards security threats of deep learning systems: A survey," *IEEE Transactions on Software Engineering*, 2020.

[257] R. Bhaskar, *Reclaiming reality: A critical introduction to contemporary philosophy.* 11 Main Street, Germantown, NY: Taylor & Francis, 2010.

[258] S. Fox and T. Do, "Getting real about big data: applying critical realism to analyse big data hype," *International Journal of Managing Projects in Business*, 2013.

[259] T. Rogers, "Critical realism and learning analytics research: epistemological implications of an ontological foundation," in *Proceedings of the fifth international conference on learning analytics and knowledge*, pp. 223–230, 2015.

[260] K. D. Miller, "Agent-based modeling and organization studies: A critical realist perspective," *Organization Studies*, vol. 36, no. 2, pp. 175–196, 2015.

[261] P. Törnberg and A. Törnberg, "The limits of computation: A philosophical critique of contemporary big data research," *Big Data & Society*, vol. 5, no. 2, 2018.

[262] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Computing Surveys*, vol. 50 (6), 2017.

[263] D. Gamez, "Progress in machine consciousness," vol. 17 (3), pp. 887–910, 2008.

[264] R. Brooks, C. Breazeal, M. Marjanović, B. Scassellati, and M. Williamson, "The cog project: Building a humanoid robot," 1998.

[265] S. Franklin, "Autonomous software agent for navy personnel work: A case study," in *Papers from 2003 AAAI Spring Symposium*, 2003.

Appendices

Appendix A

Supplementary Action Units Experiments

Figures A.1-A.12 show results of accuracy, F1-Score, precision, recall, training time mean, and classification time mean for the Binary Relevance and Label Powerset approaches applied in WiSARD and ClusWiSARD, respectively. In these figures n means the tuple size of the models. ClusWiSARD with Binary Relevance presented the best accuracy and the same network with Label Powerset presented the best F1-Score.

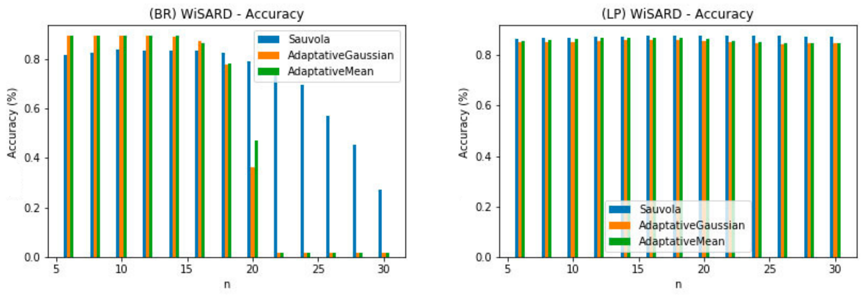


Figure A.1: Accuracy of Binary Relevance vs Label Powerset approaches using WiSARD; n = the tuple size.

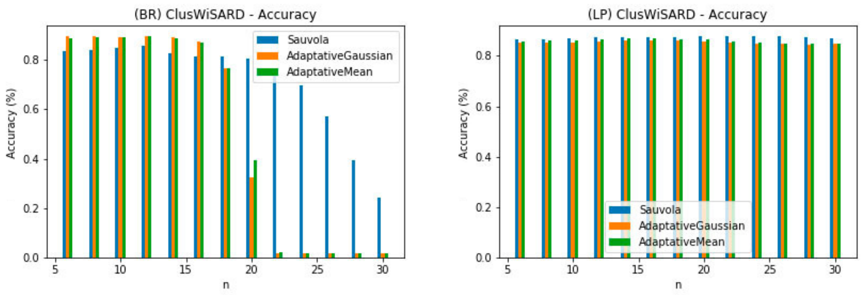


Figure A.2: Accuracy of Binary Relevance vs Label Powerset approaches using ClusWiSARD; n = the tuple size.

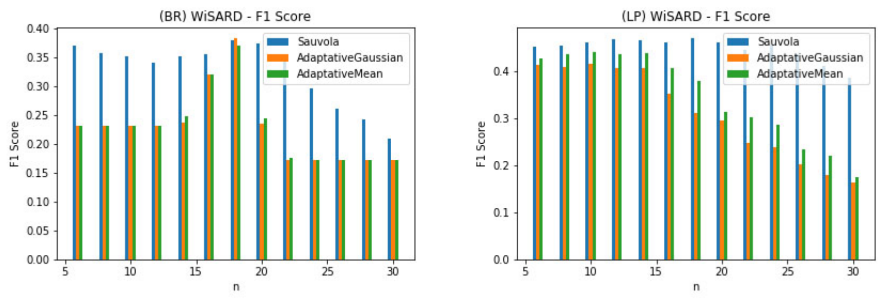


Figure A.3: F1-Score of Binary Relevance vs Label Powerset approaches using WiSARD; $n =$ the tuple size.

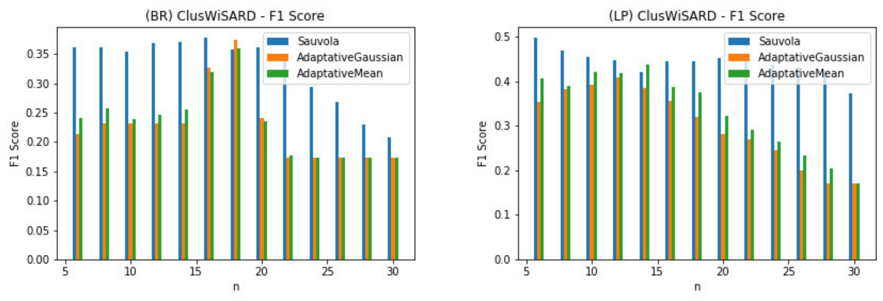


Figure A.4: F1-Score of Binary Relevance vs Label Powerset approaches using ClusWiSARD; $n =$ the tuple size.

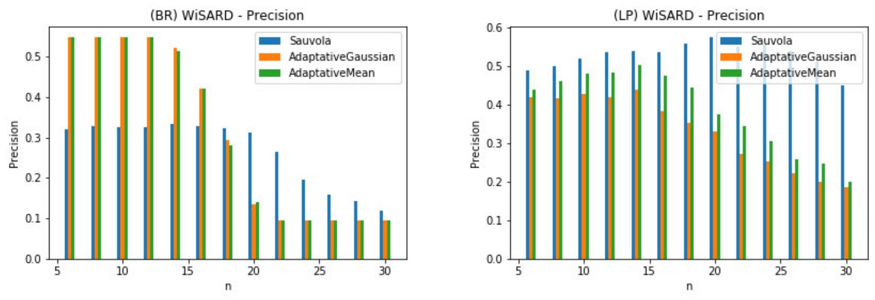


Figure A.5: Precision of Binary Relevance vs Label Powerset approaches using WiSARD; $n =$ the tuple size.

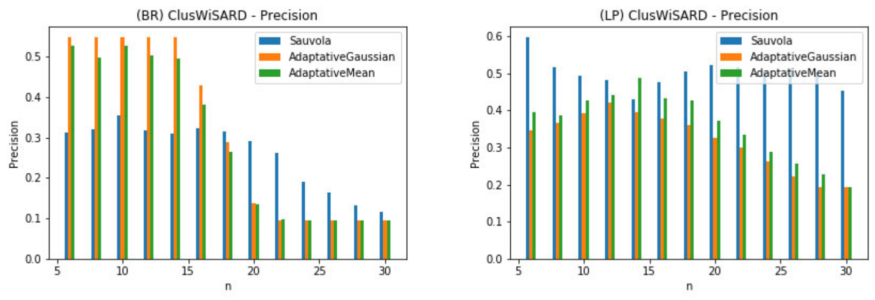


Figure A.6: Precision of Binary Relevance vs Label Powerset approaches using ClusWiSARD; $n =$ the tuple size.

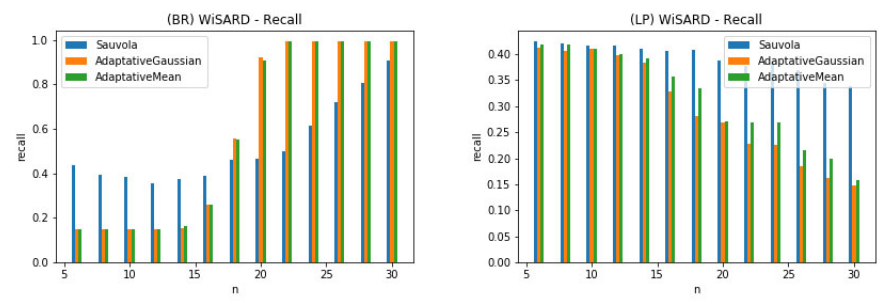


Figure A.7: Recall of Binary Relevance vs Label Powerset approaches using WiSARD; n = the tuple size.

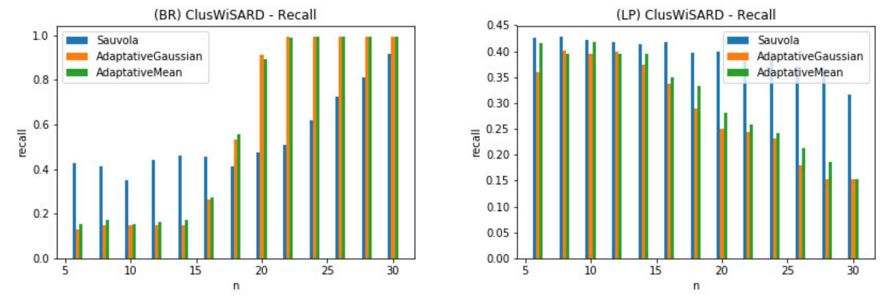


Figure A.8: Recall of Binary Relevance vs Label Powerset approaches using ClusWiSARD; n = the tuple size.

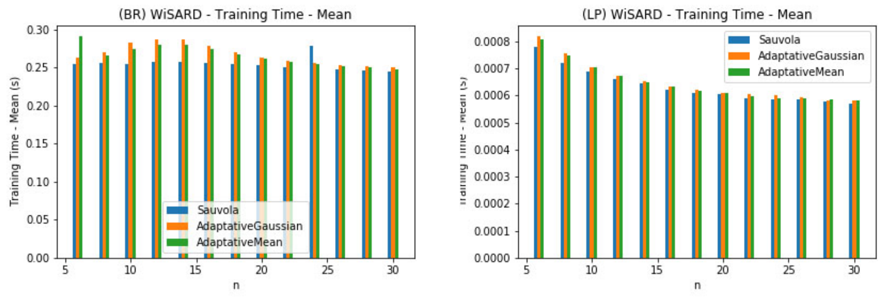


Figure A.9: Training time mean of Binary Relevance vs Label Powerset approaches using WiSARD; n = the tuple size.

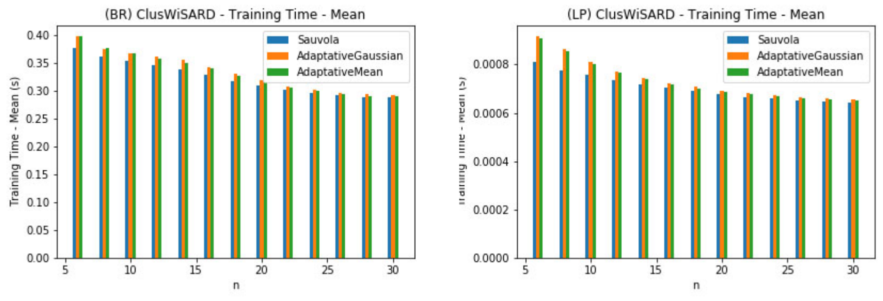


Figure A.10: Training time mean of Binary Relevance vs Label Powerset approaches using ClusWiSARD; n = the tuple size.

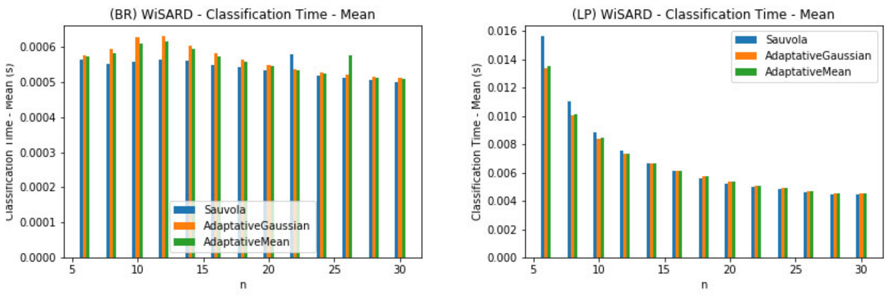


Figure A.11: Classification time mean of Binary Relevance vs Label Powerset approaches using WiSARD; n = the tuple size.

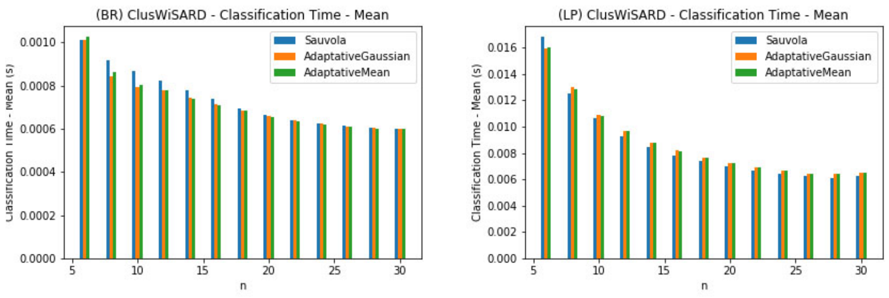


Figure A.12: Classification time mean of Binary Relevance vs Label Powerset approaches using ClusWiSARD; n = the tuple size.

The confusion matrix of best scored AUs in this result is expressed in Table A.

Table A.1: Multi-label confusion matrix for best scored classes (ClusWiSARD, Adaptive Mean, Binary Relevance): AUs 10 and 31, respectively; $M_{0,0}$ is the count of true negatives, $M_{0,1}$ is false positives, $M_{1,0}$ is false negatives and $M_{1,1}$ is true positives.

AU 10		AU 31	
347	41	360	37
182	18	169	22

Appendix B

Supplementary Ensemble Experiments

This appendix displays the complete results for all models of ensembles to the experiments reported in the Chapter 4.

The accuracy variance in all experiments described here was 0.

B.1 Cifar10 Dataset

Here are the results obtained in the Cifar10 dataset.

B.1.1 Local Threshold

The results are found in Tables B.1-B.5.

Table B.1: Results for WiSARD Bagging Ensembles in Cifar10 Dataset with Local Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.19 ± 0.01	4.35 ± 0.32	1.76e+01 ± 4.59
		Clus	0.20 ± 0.05	13.22 ± 1.92	9.61e+01 ± 44.32
		Mix	0.19 ± 0.04	11.17 ± 0.84	4.58e+01 ± 12.44
	20	WiSARD	0.17 ± 0.02	9.84 ± 0.68	3.79e+01 ± 4.86
		Clus	0.20 ± 0.05	21.99 ± 1.38	8.48e+01 ± 16.73
		Mix	0.20 ± 0.04	23.97 ± 1.34	8.45e+01 ± 19.83
0.8	10	WiSARD	0.21 ± 0.07	6.10 ± 0.74	2.36e+01 ± 1.13
		Clus	0.25 ± 0.03	17.87 ± 2.94	7.37e+01 ± 45.69
		Mix	0.23 ± 0.05	15.44 ± 1.65	5.57e+01 ± 26.24
	20	WiSARD	0.21 ± 0.07	12.93 ± 1.84	4.58e+01 ± 9.88
		Clus	0.22 ± 0.04	33.92 ± 4.20	1.28e+02 ± 31.92
		Mix	0.24 ± 0.05	33.63 ± 1.70	1.32e+02 ± 23.71

Table B.2: Results for WiSARD Boosting Ensembles in Cifar10 Dataset with Local Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.14 ± 0.00	6.05 ± 0.68	22.01 ± 9.79	8.13e+01 ± 18.69
	Clus	0.17 ± 0.04	6.05 ± 0.27	12.18 ± 3.94	5.94e+01 ± 9.00
	Mix	0.20 ± 0.06	5.92 ± 0.44	52.37 ± 56.69	1.42e+02 ± 112.79
20	WiSARD	0.17 ± 0.05	6.14 ± 0.20	39.06 ± 23.88	1.57e+02 ± 53.03
	Clus	0.16 ± 0.02	6.23 ± 0.24	36.45 ± 15.56	1.50e+02 ± 34.83
	Mix	0.15 ± 0.02	6.07 ± 0.15	62.80 ± 9.01	2.03e+02 ± 17.23

Table B.3: Results for Borda Count Ensembles in Cifar10 Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.21 ± 0.06	5.91 ± 1.22	2.81e+01 ± 1.14
		Start at 1	0.18 ± 0.02	5.85 ± 0.34	2.10e+01 ± 2.55
		Dowdall	0.22 ± 0.02	6.01 ± 0.27	2.95e+01 ± 5.91
	20	Start at 0	0.19 ± 0.04	11.79 ± 0.28	4.65e+01 ± 4.78
		Start at 1	0.21 ± 0.01	11.92 ± 0.31	5.09e+01 ± 9.83
		Dowdall	0.26 ± 0.01	12.55 ± 0.64	6.61e+01 ± 1.48
0.8	10	Start at 0	0.24 ± 0.02	7.78 ± 0.53	3.46e+01 ± 3.28
		Start at 1	0.17 ± 0.07	7.24 ± 1.13	3.26e+01 ± 4.36
		Dowdall	0.21 ± 0.06	7.91 ± 0.96	2.99e+01 ± 9.90
	20	Start at 0	0.19 ± 0.05	14.85 ± 2.17	6.24e+01 ± 8.54
		Start at 1	0.20 ± 0.03	15.92 ± 0.29	6.45e+01 ± 6.89
		Dowdall	0.18 ± 0.02	15.26 ± 0.66	5.48e+01 ± 15.38

Table B.4: Results for Tie-break Ensembles in Cifar10 Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.15 ± 0.04	5.30 ± 0.88	2.26e+01 ± 3.50
		Only Ties	0.15 ± 0.03	6.34 ± 0.43	2.58e+01 ± 1.62
		Threshold	0.14 ± 0.02	6.20 ± 0.29	2.49e+01 ± 2.96
	20	All Candidates	0.14 ± 0.03	13.65 ± 0.10	4.91e+01 ± 1.50
		Only Ties	0.15 ± 0.02	12.52 ± 0.95	5.20e+01 ± 3.83
		Threshold	0.15 ± 0.00	12.04 ± 0.32	5.19e+01 ± 3.78
0.8	10	All Candidates	0.15 ± 0.03	8.80 ± 1.19	2.82e+01 ± 2.26
		Only Ties	0.14 ± 0.01	8.59 ± 0.35	2.78e+01 ± 1.82
		Threshold	0.13 ± 0.01	8.09 ± 0.62	2.90e+01 ± 2.93
	20	All Candidates	0.16 ± 0.01	19.16 ± 0.95	5.17e+01 ± 2.02
		Only Ties	0.16 ± 0.03	18.20 ± 1.68	5.06e+01 ± 2.48
		Threshold	0.13 ± 0.01	15.72 ± 0.41	5.15e+01 ± 2.60

B.1.2 Mean Threshold

The results are found in Tables B.6-B.10.

Table B.5: Results for Weighted Votes Ensembles in Cifar10 Dataset with Local Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.16 ± 0.03	7.00 ± 0.89	2.56e+01 ± 2.85
	20	0.14 ± 0.03	12.68 ± 1.32	5.28e+01 ± 0.53
0.8	10	0.12 ± 0.00	8.18 ± 0.22	2.67e+01 ± 3.40
	20	0.13 ± 0.01	18.25 ± 2.07	4.99e+01 ± 2.98

Table B.6: Results for WiSARD Bagging Ensembles in Cifar10 Dataset with Mean Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.31 ± 0.02	4.61 ± 0.29	1.99e+01 ± 4.88
		Clus	0.33 ± 0.00	15.45 ± 1.03	5.67e+01 ± 7.21
		Mix	0.31 ± 0.01	16.11 ± 1.21	7.51e+01 ± 20.41
	20	WiSARD	0.32 ± 0.01	9.64 ± 0.77	3.77e+01 ± 4.27
		Clus	0.33 ± 0.01	29.41 ± 1.94	1.31e+02 ± 18.10
		Mix	0.33 ± 0.01	35.42 ± 6.67	1.89e+02 ± 97.75
0.8	10	WiSARD	0.33 ± 0.02	6.31 ± 1.05	2.20e+01 ± 2.15
		Clus	0.32 ± 0.01	21.09 ± 4.65	8.64e+01 ± 46.97
		Mix	0.32 ± 0.01	20.82 ± 4.75	6.77e+01 ± 15.81
	20	WiSARD	0.34 ± 0.00	13.44 ± 1.00	5.01e+01 ± 12.18
		Clus	0.33 ± 0.02	44.38 ± 8.18	1.58e+02 ± 32.97
		Mix	0.34 ± 0.01	39.49 ± 4.63	1.29e+02 ± 26.42

Table B.7: Results for WiSARD Boosting Ensembles in Cifar10 Dataset with Mean Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.26 ± 0.03	5.92 ± 0.29	22.14 ± 8.85	8.44e+01 ± 19.26
	Clus	0.29 ± 0.02	6.07 ± 0.25	18.22 ± 7.06	7.70e+01 ± 17.89
	Mix	0.26 ± 0.03	6.09 ± 0.29	16.47 ± 5.92	7.56e+01 ± 17.43
20	WiSARD	0.25 ± 0.01	6.16 ± 0.22	42.53 ± 19.57	1.71e+02 ± 32.99
	Clus	0.27 ± 0.03	6.34 ± 0.15	48.69 ± 18.93	1.89e+02 ± 40.89
	Mix	0.26 ± 0.01	6.37 ± 0.13	49.72 ± 28.83	1.90e+02 ± 60.98

B.1.3 Otsu's Binarization

The results are found in Tables B.11-B.15.

B.1.4 Yen's Binarization

The results are found in Tables B.16-B.20.

B.2 CKP Dataset

Here are the results obtained in the CKP dataset.

Table B.8: Results for Borda Count Ensembles in Cifar10 Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.31 ± 0.03	6.43 ± 0.75	3.33e+01 ± 13.31
		Start at 1	0.31 ± 0.03	6.14 ± 0.97	2.92e+01 ± 3.79
		Dowdall	0.32 ± 0.02	6.91 ± 0.66	2.96e+01 ± 5.37
	20	Start at 0	0.34 ± 0.01	12.68 ± 0.55	6.63e+01 ± 3.69
		Start at 1	0.33 ± 0.01	12.91 ± 0.61	6.93e+01 ± 14.57
		Dowdall	0.34 ± 0.01	14.58 ± 1.14	6.41e+01 ± 17.47
0.8	10	Start at 0	0.32 ± 0.01	8.52 ± 0.57	3.67e+01 ± 2.68
		Start at 1	0.33 ± 0.01	8.74 ± 0.22	3.77e+01 ± 9.43
		Dowdall	0.33 ± 0.01	8.65 ± 0.45	3.02e+01 ± 6.47
	20	Start at 0	0.34 ± 0.01	16.66 ± 1.30	7.52e+01 ± 10.30
		Start at 1	0.32 ± 0.01	15.79 ± 0.94	5.81e+01 ± 5.74
		Dowdall	0.33 ± 0.01	16.15 ± 0.83	5.74e+01 ± 13.72

Table B.9: Results for Tie-break Ensembles in Cifar10 Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.19 ± 0.01	6.43 ± 0.59	2.36e+01 ± 3.25
		Only Ties	0.20 ± 0.02	6.36 ± 0.45	2.71e+01 ± 2.06
		Threshold	0.18 ± 0.02	6.35 ± 0.53	2.47e+01 ± 1.99
	20	All Candidates	0.21 ± 0.02	13.05 ± 1.32	5.38e+01 ± 4.99
		Only Ties	0.21 ± 0.03	14.89 ± 0.71	5.00e+01 ± 5.32
		Threshold	0.19 ± 0.03	11.33 ± 0.12	4.60e+01 ± 3.42
0.8	10	All Candidates	0.20 ± 0.01	8.73 ± 0.58	2.79e+01 ± 0.77
		Only Ties	0.21 ± 0.02	8.08 ± 0.06	2.51e+01 ± 3.46
		Threshold	0.18 ± 0.01	8.19 ± 0.68	2.70e+01 ± 2.21
	20	All Candidates	0.20 ± 0.00	17.43 ± 0.62	5.48e+01 ± 2.25
		Only Ties	0.23 ± 0.01	17.32 ± 1.31	4.67e+01 ± 2.25
		Threshold	0.20 ± 0.01	15.89 ± 0.98	5.01e+01 ± 3.18

Table B.10: Results for Weighted Votes Ensembles in Cifar10 Dataset with Mean Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.21 ± 0.00	6.65 ± 0.23	2.62e+01 ± 1.10
	20	0.20 ± 0.01	11.19 ± 0.54	5.03e+01 ± 4.70
0.8	10	0.20 ± 0.01	7.48 ± 0.55	2.32e+01 ± 0.73
	20	0.21 ± 0.02	17.65 ± 1.59	4.98e+01 ± 0.91

B.2.1 Local Threshold

The results are found in Tables B.21-B.25.

B.2.2 Mean Threshold

The results are found in Tables B.26-B30.

Table B.11: Results for WiSARD Bagging Ensembles in Cifar10 Dataset with Otsu's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.34 ± 0.01	4.67 ± 0.48	2.23e+01 ± 0.59
		Clus	0.34 ± 0.00	12.72 ± 0.28	5.16e+01 ± 13.59
		Mix	0.34 ± 0.01	15.06 ± 2.41	6.44e+01 ± 25.19
	20	WiSARD	0.35 ± 0.00	9.55 ± 0.33	4.33e+01 ± 2.41
		Clus	0.36 ± 0.01	37.05 ± 4.61	1.75e+02 ± 37.38
		Mix	0.34 ± 0.01	30.55 ± 4.34	1.33e+02 ± 48.20
0.8	10	WiSARD	0.35 ± 0.01	6.52 ± 0.98	2.64e+01 ± 3.97
		Clus	0.34 ± 0.01	25.54 ± 3.57	1.07e+02 ± 43.50
		Mix	0.32 ± 0.01	31.27 ± 6.25	1.76e+02 ± 53.04
	20	WiSARD	0.35 ± 0.00	12.18 ± 0.76	3.57e+01 ± 5.43
		Clus	0.36 ± 0.00	41.61 ± 1.05	1.48e+02 ± 22.34
		Mix	0.36 ± 0.00	44.19 ± 6.87	1.59e+02 ± 40.37

Table B.12: Results for WiSARD Boosting Ensembles in Cifar10 Dataset with Otsu's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.29 ± 0.01	6.03 ± 0.12	22.72 ± 7.15	9.09e+01 ± 16.39
	Clus	0.31 ± 0.00	6.07 ± 0.19	31.29 ± 24.22	1.05e+02 ± 45.63
	Mix	0.30 ± 0.01	5.90 ± 0.31	32.58 ± 26.58	1.05e+02 ± 53.36
20	WiSARD	0.29 ± 0.01	6.05 ± 0.02	35.66 ± 11.32	1.63e+02 ± 27.74
	Clus	0.29 ± 0.02	6.52 ± 0.26	58.79 ± 26.45	2.12e+02 ± 52.85
	Mix	0.28 ± 0.01	6.20 ± 0.18	46.23 ± 17.33	1.88e+02 ± 40.57

Table B.13: Results for Borda Count ensembles in Cifar10 dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.34 ± 0.01	6.12 ± 0.35	3.05e+01 ± 5.60
		Start at 1	0.34 ± 0.01	6.39 ± 0.23	2.85e+01 ± 5.87
		Dowdall	0.31 ± 0.01	5.53 ± 0.41	4.14e+01 ± 4.14
	20	Start at 0	0.36 ± 0.00	13.15 ± 0.66	6.27e+01 ± 0.89
		Start at 1	0.35 ± 0.01	12.97 ± 1.79	6.85e+01 ± 14.09
		Dowdall	0.34 ± 0.02	12.65 ± 0.26	5.72e+01 ± 8.77
0.8	10	Start at 0	0.34 ± 0.01	8.46 ± 0.49	3.63e+01 ± 1.73
		Start at 1	0.33 ± 0.00	7.80 ± 0.39	4.51e+01 ± 11.56
		Dowdall	0.33 ± 0.01	8.29 ± 0.17	3.70e+01 ± 9.26
	20	Start at 0	0.35 ± 0.01	16.41 ± 1.75	6.74e+01 ± 2.78
		Start at 1	0.35 ± 0.01	15.53 ± 1.07	6.73e+01 ± 13.07
		Dowdall	0.35 ± 0.01	14.89 ± 0.53	5.71e+01 ± 10.71

B.2.3 Otsu's Binarization

The results are found in Tables B.31-B.35.

Table B.14: Results for Tie-break Ensembles in Cifar10 Dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.21 ± 0.01	5.34 ± 0.28	2.34e+01 ± 2.24
		Only Ties	0.20 ± 0.01	5.50 ± 0.61	2.35e+01 ± 4.82
		Threshold	0.18 ± 0.01	6.14 ± 0.19	2.56e+01 ± 3.56
	20	All Candidates	0.23 ± 0.02	11.53 ± 1.02	4.70e+01 ± 3.62
		Only Ties	0.23 ± 0.02	11.05 ± 0.97	4.61e+01 ± 0.31
		Threshold	0.20 ± 0.01	11.59 ± 0.71	4.93e+01 ± 6.05
0.8	10	All Candidates	0.21 ± 0.02	7.77 ± 0.97	2.54e+01 ± 3.60
		Only Ties	0.20 ± 0.01	7.62 ± 0.06	2.59e+01 ± 2.76
		Threshold	0.17 ± 0.02	7.79 ± 0.50	2.49e+01 ± 1.10
	20	All Candidates	0.24 ± 0.00	14.70 ± 0.65	4.99e+01 ± 1.82
		Only Ties	0.24 ± 0.01	15.20 ± 1.10	4.65e+01 ± 2.45
		Threshold	0.20 ± 0.01	15.28 ± 0.55	5.00e+01 ± 3.33

Table B.15: Results for Weighted Votes Ensembles in Cifar10 Dataset with Otsu's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.20 ± 0.01	5.56 ± 0.11	2.41e+01 ± 3.21
	20	0.24 ± 0.02	11.06 ± 0.25	4.31e+01 ± 1.40
0.8	10	0.21 ± 0.01	7.96 ± 0.39	2.51e+01 ± 2.32
	20	0.23 ± 0.01	16.20 ± 0.74	4.72e+01 ± 4.04

Table B.16: Results for WiSARD Bagging Ensembles in Cifar10 Dataset with Yen's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.33 ± 0.00	4.01 ± 0.37	1.85e+01 ± 3.81
		Clus	0.32 ± 0.01	15.61 ± 7.41	7.03e+01 ± 46.86
		Mix	0.32 ± 0.01	15.74 ± 4.00	7.16e+01 ± 34.46
	20	WiSARD	0.34 ± 0.01	7.72 ± 0.67	3.64e+01 ± 12.10
		Clus	0.33 ± 0.01	34.35 ± 6.12	1.56e+02 ± 23.56
		Mix	0.34 ± 0.01	37.20 ± 4.98	1.74e+02 ± 18.31
0.8	10	WiSARD	0.34 ± 0.01	5.35 ± 0.43	2.27e+01 ± 6.29
		Clus	0.33 ± 0.00	18.09 ± 2.80	6.61e+01 ± 11.93
		Mix	0.33 ± 0.01	21.08 ± 1.80	8.35e+01 ± 31.95
	20	WiSARD	0.35 ± 0.00	13.11 ± 0.77	4.54e+01 ± 0.07
		Clus	0.34 ± 0.01	45.18 ± 4.61	1.43e+02 ± 33.99
		Mix	0.34 ± 0.00	53.28 ± 7.83	1.91e+02 ± 36.35

B.2.4 Yen's Binarization

The results are found in Tables B.36-B.40.

Table B.17: Results for WiSARD Boosting Ensembles in Cifar10 Dataset with Yen's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.30 ± 0.00	5.95 ± 0.72	19.22 ± 9.15	8.30e+01 ± 19.60
	Clus	0.30 ± 0.01	5.81 ± 0.28	18.20 ± 9.25	8.02e+01 ± 26.20
	Mix	0.29 ± 0.01	5.81 ± 0.12	21.05 ± 1.72	8.52e+01 ± 2.70
20	WiSARD	0.27 ± 0.01	5.90 ± 0.23	33.19 ± 25.36	1.54e+02 ± 55.02
	Clus	0.28 ± 0.01	6.13 ± 0.19	36.05 ± 13.26	1.62e+02 ± 28.51
	Mix	0.29 ± 0.01	6.46 ± 0.19	60.20 ± 28.92	2.08e+02 ± 55.95

Table B.18: Results for Borda Count Ensembles in Cifar10 Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.33 ± 0.00	6.22 ± 0.21	2.85e+01 ± 1.17
		Start at 1	0.33 ± 0.00	5.63 ± 0.25	3.24e+01 ± 2.17
		Dowdall	0.32 ± 0.00	5.43 ± 0.32	3.18e+01 ± 9.68
	20	Start at 0	0.34 ± 0.00	10.66 ± 0.45	6.00e+01 ± 10.77
		Start at 1	0.34 ± 0.01	10.95 ± 0.73	5.63e+01 ± 10.17
		Dowdall	0.34 ± 0.01	11.09 ± 0.76	4.60e+01 ± 0.65
0.8	10	Start at 0	0.32 ± 0.01	7.49 ± 0.50	3.25e+01 ± 9.15
		Start at 1	0.32 ± 0.01	6.89 ± 0.29	3.84e+01 ± 12.63
		Dowdall	0.33 ± 0.00	7.53 ± 0.26	2.66e+01 ± 5.54
	20	Start at 0	0.34 ± 0.01	14.31 ± 0.20	6.71e+01 ± 12.97
		Start at 1	0.35 ± 0.00	15.32 ± 0.71	6.12e+01 ± 9.94
		Dowdall	0.34 ± 0.00	14.30 ± 0.86	5.53e+01 ± 10.30

Table B.19: Results for Tie-break Ensembles in Cifar10 Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.21 ± 0.01	5.66 ± 0.43	2.19e+01 ± 1.30
		Only Ties	0.20 ± 0.01	6.06 ± 0.36	2.55e+01 ± 1.47
		Threshold	0.18 ± 0.01	5.76 ± 0.82	2.41e+01 ± 2.70
	20	All Candidates	0.21 ± 0.01	10.94 ± 0.51	5.13e+01 ± 7.13
		Only Ties	0.22 ± 0.01	11.50 ± 0.05	4.90e+01 ± 5.06
		Threshold	0.19 ± 0.01	11.47 ± 1.04	5.00e+01 ± 5.74
0.8	10	All Candidates	0.20 ± 0.01	7.93 ± 0.31	2.83e+01 ± 2.58
		Only Ties	0.21 ± 0.00	7.90 ± 0.72	2.48e+01 ± 2.82
		Threshold	0.18 ± 0.00	7.84 ± 0.20	2.55e+01 ± 2.96
	20	All Candidates	0.23 ± 0.01	15.01 ± 0.80	4.92e+01 ± 5.81
		Only Ties	0.23 ± 0.01	15.32 ± 0.75	4.94e+01 ± 3.19
		Threshold	0.19 ± 0.01	14.88 ± 0.47	4.84e+01 ± 2.37

B.3 Fashion MNIST Dataset

Here are the results obtained in the Fashion MNIST dataset.

Table B.20: Results for Weighted Votes Ensembles in Cifar10 Dataset with Yen's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.21 ± 0.01	5.98 ± 0.40	2.41e+01 ± 3.40
	20	0.22 ± 0.01	11.10 ± 0.50	4.99e+01 ± 5.66
0.8	10	0.22 ± 0.01	7.85 ± 0.37	2.50e+01 ± 0.88
	20	0.23 ± 0.02	15.60 ± 0.42	4.77e+01 ± 2.31

Table B.21: Results for WiSARD Bagging Ensembles in CKP Dataset with Local Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.37 ± 0.02	0.15 ± 0.02	3.08e-01 ± 0.08
		Clus	0.31 ± 0.07	0.46 ± 0.02	9.29e-01 ± 0.04
		Mix	0.38 ± 0.03	0.54 ± 0.06	1.10e+00 ± 0.12
	20	WiSARD	0.44 ± 0.02	0.28 ± 0.04	4.92e-01 ± 0.06
		Clus	0.34 ± 0.05	0.93 ± 0.10	1.93e+00 ± 0.29
		Mix	0.39 ± 0.04	1.00 ± 0.13	2.05e+00 ± 0.31
0.8	10	WiSARD	0.43 ± 0.03	0.18 ± 0.01	3.03e-01 ± 0.03
		Clus	0.31 ± 0.07	0.62 ± 0.19	1.06e+00 ± 0.41
		Mix	0.40 ± 0.03	0.62 ± 0.04	1.04e+00 ± 0.07
	20	WiSARD	0.43 ± 0.02	0.40 ± 0.03	8.20e-01 ± 0.10
		Clus	0.35 ± 0.02	1.32 ± 0.17	2.30e+00 ± 0.35
		Mix	0.39 ± 0.03	1.20 ± 0.14	2.06e+00 ± 0.30

Table B.22: Results for WiSARD Boosting Ensembles in CKP Dataset with Local Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.37 ± 0.01	0.08 ± 0.00	0.13 ± 0.00	2.23e+00 ± 0.01
	Clus	0.34 ± 0.07	0.08 ± 0.02	0.13 ± 0.03	2.24e+00 ± 0.08
	Mix	0.27 ± 0.03	0.09 ± 0.01	0.15 ± 0.02	2.27e+00 ± 0.05
20	WiSARD	0.27 ± 0.02	0.09 ± 0.01	0.16 ± 0.01	4.27e+00 ± 0.10
	Clus	0.26 ± 0.02	0.10 ± 0.02	0.17 ± 0.03	4.30e+00 ± 0.13
	Mix	0.26 ± 0.02	0.09 ± 0.01	0.17 ± 0.01	4.32e+00 ± 0.03

B.3.1 Local Threshold

The results are found in Tables B.41 - B.45.

B.3.2 Mean Threshold

The results are found in Tables B.46-B.50.

B.3.3 Otsu's Binarization

The results are found in Tables B.51-B.55.

Table B.23: Results for Borda Count Ensembles in CKP Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.34 ± 0.04	0.19 ± 0.02	6.48e-01 ± 0.09
		Start at 1	0.39 ± 0.02	0.20 ± 0.02	6.97e-01 ± 0.06
		Dowdall	0.34 ± 0.06	0.19 ± 0.01	6.49e-01 ± 0.05
	20	Start at 0	0.40 ± 0.01	0.40 ± 0.01	1.20e+00 ± 0.05
		Start at 1	0.37 ± 0.01	0.38 ± 0.03	1.13e+00 ± 0.11
		Dowdall	0.37 ± 0.02	0.40 ± 0.04	1.19e+00 ± 0.13
0.8	10	Start at 0	0.40 ± 0.03	0.25 ± 0.02	7.15e-01 ± 0.04
		Start at 1	0.41 ± 0.04	0.25 ± 0.02	7.50e-01 ± 0.07
		Dowdall	0.43 ± 0.02	0.31 ± 0.05	9.29e-01 ± 0.17
	20	Start at 0	0.40 ± 0.05	0.47 ± 0.06	1.19e+00 ± 0.14
		Start at 1	0.45 ± 0.03	0.55 ± 0.03	1.39e+00 ± 0.09
		Dowdall	0.43 ± 0.02	0.56 ± 0.07	1.45e+00 ± 0.19

Table B.24: Results for Tie-break Ensembles in CKP Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.51 ± 0.03	0.21 ± 0.01	6.60e-01 ± 0.04
		Only Ties	0.55 ± 0.08	0.21 ± 0.02	6.83e-01 ± 0.06
		Threshold	0.41 ± 0.21	0.22 ± 0.02	7.02e-01 ± 0.05
	20	All Candidates	0.34 ± 0.15	0.42 ± 0.04	1.22e+00 ± 0.12
		Only Ties	0.30 ± 0.22	0.39 ± 0.04	1.14e+00 ± 0.13
		Threshold	0.43 ± 0.21	0.39 ± 0.03	1.12e+00 ± 0.08
0.8	10	All Candidates	0.39 ± 0.14	0.27 ± 0.00	7.48e-01 ± 0.02
		Only Ties	0.39 ± 0.20	0.26 ± 0.02	7.16e-01 ± 0.04
		Threshold	0.30 ± 0.15	0.22 ± 0.01	6.15e-01 ± 0.03
	20	All Candidates	0.47 ± 0.15	0.59 ± 0.06	1.45e+00 ± 0.15
		Only Ties	0.30 ± 0.21	0.51 ± 0.02	1.25e+00 ± 0.04
		Threshold	0.44 ± 0.11	0.53 ± 0.04	1.31e+00 ± 0.09

Table B.25: Results for Weighted Votes Ensembles in CKP Dataset with Local Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.27 ± 0.09	0.17 ± 0.03	5.77e-01 ± 0.09
	20	0.32 ± 0.20	0.39 ± 0.03	1.12e+00 ± 0.09
0.8	10	0.42 ± 0.21	0.26 ± 0.02	7.20e-01 ± 0.04
	20	0.38 ± 0.17	0.50 ± 0.02	1.22e+00 ± 0.06

Table B.26: Results for WiSARD Bagging Ensembles in CKP Dataset with Mean Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.55 ± 0.02	0.15 ± 0.01	3.17e-01 ± 0.03
		Clus	0.53 ± 0.00	0.47 ± 0.08	1.01e+00 ± 0.20
		Mix	0.53 ± 0.02	0.51 ± 0.08	1.09e+00 ± 0.18
	20	WiSARD	0.54 ± 0.01	0.28 ± 0.00	5.30e-01 ± 0.02
		Clus	0.54 ± 0.03	0.97 ± 0.04	2.04e+00 ± 0.11
		Mix	0.55 ± 0.02	0.83 ± 0.03	1.61e+00 ± 0.10
0.8	10	WiSARD	0.56 ± 0.01	0.18 ± 0.02	3.11e-01 ± 0.04
		Clus	0.52 ± 0.02	0.63 ± 0.08	1.12e+00 ± 0.18
		Mix	0.54 ± 0.02	0.63 ± 0.08	1.15e+00 ± 0.14
	20	WiSARD	0.55 ± 0.02	0.34 ± 0.04	5.38e-01 ± 0.11
		Clus	0.56 ± 0.00	1.18 ± 0.18	2.02e+00 ± 0.39
		Mix	0.54 ± 0.02	1.14 ± 0.06	1.93e+00 ± 0.13

Table B.27: Results for WiSARD Boosting Ensembles in CKP Dataset with Mean Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.51 ± 0.01	0.07 ± 0.02	0.11 ± 0.03	2.22e+00 ± 0.07
	Clus	0.45 ± 0.04	0.07 ± 0.01	0.12 ± 0.02	2.25e+00 ± 0.04
	Mix	0.48 ± 0.03	0.08 ± 0.01	0.14 ± 0.01	2.28e+00 ± 0.03
20	WiSARD	0.43 ± 0.04	0.09 ± 0.01	0.16 ± 0.02	4.24e+00 ± 0.08
	Clus	0.44 ± 0.02	0.08 ± 0.01	0.15 ± 0.02	4.03e+00 ± 0.07
	Mix	0.46 ± 0.06	0.10 ± 0.00	0.17 ± 0.01	4.10e+00 ± 0.02

Table B.28: Results for Borda Count Ensembles in CKP Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.53 ± 0.02	0.18 ± 0.01	6.48e-01 ± 0.03
		Start at 1	0.52 ± 0.03	0.17 ± 0.01	6.03e-01 ± 0.03
		Dowdall	0.55 ± 0.04	0.21 ± 0.02	7.25e-01 ± 0.06
	20	Start at 0	0.55 ± 0.01	0.36 ± 0.02	1.11e+00 ± 0.08
		Start at 1	0.55 ± 0.03	0.36 ± 0.02	1.08e+00 ± 0.04
		Dowdall	0.56 ± 0.00	0.39 ± 0.01	1.18e+00 ± 0.05
0.8	10	Start at 0	0.54 ± 0.02	0.25 ± 0.04	7.64e-01 ± 0.13
		Start at 1	0.55 ± 0.01	0.25 ± 0.01	7.58e-01 ± 0.01
		Dowdall	0.56 ± 0.02	0.26 ± 0.00	7.80e-01 ± 0.00
	20	Start at 0	0.56 ± 0.00	0.43 ± 0.00	1.07e+00 ± 0.02
		Start at 1	0.56 ± 0.01	0.47 ± 0.02	1.21e+00 ± 0.06
		Dowdall	0.54 ± 0.01	0.44 ± 0.01	1.10e+00 ± 0.02

Table B.29: Results for Tie-break Ensembles in CKP Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.56 ± 0.06	0.19 ± 0.01	6.04e-01 ± 0.03
		Only Ties	0.59 ± 0.03	0.19 ± 0.01	6.21e-01 ± 0.04
		Threshold	0.53 ± 0.02	0.18 ± 0.01	5.96e-01 ± 0.02
	20	All Candidates	0.61 ± 0.10	0.38 ± 0.00	1.10e+00 ± 0.01
		Only Ties	0.64 ± 0.01	0.40 ± 0.01	1.12e+00 ± 0.02
		Threshold	0.58 ± 0.07	0.37 ± 0.04	1.06e+00 ± 0.11
0.8	10	All Candidates	0.58 ± 0.07	0.23 ± 0.02	6.37e-01 ± 0.05
		Only Ties	0.58 ± 0.08	0.24 ± 0.02	6.71e-01 ± 0.04
		Threshold	0.59 ± 0.04	0.24 ± 0.02	6.52e-01 ± 0.05
	20	All Candidates	0.63 ± 0.01	0.48 ± 0.05	1.15e+00 ± 0.11
		Only Ties	0.60 ± 0.04	0.45 ± 0.04	1.10e+00 ± 0.10
		Threshold	0.61 ± 0.03	0.47 ± 0.03	1.14e+00 ± 0.08

Table B.30: Results for Weighted Votes Ensembles in CKP Dataset with Mean Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.60 ± 0.06	0.20 ± 0.03	6.43e-01 ± 0.07
	20	0.59 ± 0.03	0.38 ± 0.01	1.08e+00 ± 0.02
0.8	10	0.61 ± 0.02	0.23 ± 0.03	6.60e-01 ± 0.07
	20	0.60 ± 0.04	0.47 ± 0.01	1.14e+00 ± 0.03

Table B.31: Results for WiSARD Bagging Ensembles in CKP Dataset with Otsu's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.51 ± 0.02	0.12 ± 0.01	2.05e-01 ± 0.03
		Clus	0.48 ± 0.05	0.43 ± 0.06	8.85e-01 ± 0.17
		Mix	0.51 ± 0.05	0.46 ± 0.03	9.43e-01 ± 0.06
	20	WiSARD	0.54 ± 0.01	0.27 ± 0.03	5.07e-01 ± 0.11
		Clus	0.50 ± 0.00	0.94 ± 0.09	2.04e+00 ± 0.26
		Mix	0.53 ± 0.03	0.96 ± 0.06	2.04e+00 ± 0.20
0.8	10	WiSARD	0.53 ± 0.01	0.18 ± 0.02	2.88e-01 ± 0.02
		Clus	0.47 ± 0.04	0.57 ± 0.07	9.97e-01 ± 0.15
		Mix	0.49 ± 0.04	0.55 ± 0.03	9.08e-01 ± 0.06
	20	WiSARD	0.53 ± 0.01	0.37 ± 0.03	6.52e-01 ± 0.07
		Clus	0.49 ± 0.02	1.15 ± 0.09	2.00e+00 ± 0.21
		Mix	0.51 ± 0.01	1.22 ± 0.11	2.11e+00 ± 0.23

Table B.32: Results for WiSARD Boosting Ensembles in CKP Dataset with Otsu's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.47 ± 0.03	0.08 ± 0.01	0.13 ± 0.01	2.15e+00 ± 0.04
	Clus	0.47 ± 0.04	0.08 ± 0.01	0.13 ± 0.01	2.17e+00 ± 0.02
	Mix	0.46 ± 0.03	0.06 ± 0.01	0.08 ± 0.01	2.05e+00 ± 0.03
20	WiSARD	0.46 ± 0.01	0.09 ± 0.00	0.15 ± 0.01	4.08e+00 ± 0.06
	Clus	0.43 ± 0.06	0.10 ± 0.01	0.17 ± 0.01	4.19e+00 ± 0.03
	Mix	0.43 ± 0.06	0.10 ± 0.01	0.17 ± 0.01	4.20e+00 ± 0.03

Table B.33: Results for Borda Count Ensembles in CKP Dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.49 ± 0.01	0.20 ± 0.02	6.85e-01 ± 0.07
		Start at 1	0.51 ± 0.02	0.19 ± 0.00	6.44e-01 ± 0.02
		Dowdall	0.51 ± 0.03	0.19 ± 0.01	6.61e-01 ± 0.06
	20	Start at 0	0.54 ± 0.02	0.35 ± 0.04	1.04e+00 ± 0.14
		Start at 1	0.54 ± 0.02	0.38 ± 0.03	1.14e+00 ± 0.08
		Dowdall	0.54 ± 0.01	0.39 ± 0.00	1.17e+00 ± 0.01
0.8	10	Start at 0	0.49 ± 0.02	0.24 ± 0.01	6.70e-01 ± 0.04
		Start at 1	0.52 ± 0.00	0.25 ± 0.00	7.22e-01 ± 0.01
		Dowdall	0.51 ± 0.01	0.24 ± 0.02	6.87e-01 ± 0.06
	20	Start at 0	0.52 ± 0.00	0.49 ± 0.04	1.24e+00 ± 0.11
		Start at 1	0.52 ± 0.02	0.49 ± 0.04	1.25e+00 ± 0.12
		Dowdall	0.51 ± 0.02	0.52 ± 0.03	1.35e+00 ± 0.08

Table B.34: Results for Tie-break Ensembles in CKP Dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.53 ± 0.03	0.19 ± 0.03	5.98e-01 ± 0.07
		Only Ties	0.53 ± 0.05	0.19 ± 0.03	5.90e-01 ± 0.08
		Threshold	0.48 ± 0.05	0.20 ± 0.03	6.16e-01 ± 0.05
	20	All Candidates	0.53 ± 0.06	0.38 ± 0.01	1.06e+00 ± 0.04
		Only Ties	0.57 ± 0.03	0.37 ± 0.03	1.06e+00 ± 0.11
		Threshold	0.55 ± 0.02	0.38 ± 0.01	1.08e+00 ± 0.04
0.8	10	All Candidates	0.47 ± 0.09	0.21 ± 0.05	5.81e-01 ± 0.10
		Only Ties	0.53 ± 0.02	0.25 ± 0.03	6.82e-01 ± 0.06
		Threshold	0.46 ± 0.05	0.22 ± 0.03	5.97e-01 ± 0.05
	20	All Candidates	0.54 ± 0.02	0.50 ± 0.05	1.21e+00 ± 0.11
		Only Ties	0.53 ± 0.03	0.48 ± 0.02	1.17e+00 ± 0.04
		Threshold	0.51 ± 0.02	0.49 ± 0.01	1.20e+00 ± 0.03

Table B.35: Results for Weighted Votes Ensembles in CKP Dataset with Otsu's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.56 ± 0.01	0.19 ± 0.01	5.94e-01 ± 0.03
	20	0.58 ± 0.04	0.38 ± 0.01	1.08e+00 ± 0.03
0.8	10	0.55 ± 0.03	0.25 ± 0.00	6.65e-01 ± 0.02
	20	0.54 ± 0.02	0.50 ± 0.03	1.20e+00 ± 0.07

Table B.36: Results for WiSARD Bagging Ensembles in CKP Dataset with Yen's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.39 ± 0.01	0.15 ± 0.02	3.18e-01 ± 0.08
		Clus	0.39 ± 0.03	0.46 ± 0.01	9.73e-01 ± 0.05
		Mix	0.41 ± 0.05	0.44 ± 0.01	8.88e-01 ± 0.02
	20	WiSARD	0.43 ± 0.02	0.29 ± 0.03	5.47e-01 ± 0.06
		Clus	0.42 ± 0.00	0.90 ± 0.07	1.89e+00 ± 0.18
		Mix	0.40 ± 0.02	0.93 ± 0.05	1.87e+00 ± 0.13
0.8	10	WiSARD	0.41 ± 0.03	0.17 ± 0.01	3.08e-01 ± 0.03
		Clus	0.36 ± 0.05	0.65 ± 0.03	1.17e+00 ± 0.07
		Mix	0.40 ± 0.02	0.60 ± 0.12	1.07e+00 ± 0.29
	20	WiSARD	0.43 ± 0.01	0.34 ± 0.02	5.68e-01 ± 0.06
		Clus	0.39 ± 0.02	1.17 ± 0.08	2.03e+00 ± 0.15
		Mix	0.40 ± 0.01	1.21 ± 0.04	2.09e+00 ± 0.09

Table B.37: Results for WiSARD Boosting Ensembles in CKP Dataset with Yen's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.38 ± 0.03	0.08 ± 0.01	0.13 ± 0.02	2.26e+00 ± 0.04
	Clus	0.36 ± 0.01	0.09 ± 0.02	0.16 ± 0.03	2.33e+00 ± 0.07
	Mix	0.39 ± 0.02	0.08 ± 0.01	0.13 ± 0.02	2.26e+00 ± 0.05
20	WiSARD	0.36 ± 0.02	0.09 ± 0.01	0.17 ± 0.01	4.26e+00 ± 0.08
	Clus	0.34 ± 0.02	0.10 ± 0.01	0.17 ± 0.02	4.28e+00 ± 0.07
	Mix	0.35 ± 0.02	0.09 ± 0.01	0.16 ± 0.02	4.28e+00 ± 0.03

Table B.38: Results for Borda Count Ensembles in CKP Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.39 ± 0.02	0.19 ± 0.03	6.63e-01 ± 0.09
		Start at 1	0.39 ± 0.02	0.19 ± 0.03	6.68e-01 ± 0.07
		Dowdall	0.40 ± 0.01	0.18 ± 0.01	6.51e-01 ± 0.05
	20	Start at 0	0.42 ± 0.02	0.39 ± 0.03	1.15e+00 ± 0.05
		Start at 1	0.42 ± 0.02	0.39 ± 0.02	1.15e+00 ± 0.09
		Dowdall	0.42 ± 0.02	0.36 ± 0.03	1.09e+00 ± 0.12
0.8	10	Start at 0	0.42 ± 0.03	0.27 ± 0.03	7.89e-01 ± 0.09
		Start at 1	0.41 ± 0.01	0.24 ± 0.02	7.30e-01 ± 0.06
		Dowdall	0.41 ± 0.02	0.25 ± 0.03	7.31e-01 ± 0.06
	20	Start at 0	0.42 ± 0.02	0.47 ± 0.03	1.18e+00 ± 0.09
		Start at 1	0.44 ± 0.00	0.48 ± 0.02	1.25e+00 ± 0.07
		Dowdall	0.44 ± 0.03	0.46 ± 0.01	1.18e+00 ± 0.03

Table B.39: Results for Tie-break Ensembles in CKP Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.41 ± 0.06	0.20 ± 0.02	6.53e-01 ± 0.06
		Only Ties	0.39 ± 0.03	0.18 ± 0.01	6.08e-01 ± 0.03
		Threshold	0.37 ± 0.03	0.19 ± 0.01	6.35e-01 ± 0.02
	20	All Candidates	0.41 ± 0.07	0.40 ± 0.05	1.12e+00 ± 0.13
		Only Ties	0.42 ± 0.06	0.38 ± 0.01	1.09e+00 ± 0.04
		Threshold	0.43 ± 0.04	0.38 ± 0.00	1.06e+00 ± 0.00
0.8	10	All Candidates	0.38 ± 0.04	0.25 ± 0.03	6.81e-01 ± 0.07
		Only Ties	0.43 ± 0.07	0.26 ± 0.03	6.96e-01 ± 0.05
		Threshold	0.39 ± 0.02	0.23 ± 0.01	6.53e-01 ± 0.02
	20	All Candidates	0.40 ± 0.06	0.47 ± 0.02	1.13e+00 ± 0.05
		Only Ties	0.41 ± 0.02	0.47 ± 0.02	1.12e+00 ± 0.04
		Threshold	0.37 ± 0.02	0.46 ± 0.01	1.11e+00 ± 0.03

Table B.40: Results for Weighted Votes Ensembles in CKP Dataset with Yen's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.41 ± 0.02	0.20 ± 0.01	6.43e-01 ± 0.04
	20	0.42 ± 0.03	0.39 ± 0.00	1.08e+00 ± 0.00
0.8	10	0.39 ± 0.05	0.25 ± 0.03	6.85e-01 ± 0.06
	20	0.42 ± 0.02	0.50 ± 0.03	1.20e+00 ± 0.07

Table B.41: Results for WiSARD Bagging Ensembles in Fashion MNIST Dataset with Local Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.82 ± 0.01	2.28 ± 0.25	1.32e+01 ± 1.55
		Clus	0.83 ± 0.02	16.13 ± 1.78	9.21e+01 ± 51.34
		Mix	0.83 ± 0.00	14.00 ± 2.27	6.89e+01 ± 19.36
	20	WiSARD	0.83 ± 0.00	4.71 ± 0.02	2.53e+01 ± 7.39
		Clus	0.84 ± 0.00	33.06 ± 2.59	1.76e+02 ± 21.79
		Mix	0.84 ± 0.00	27.21 ± 2.64	1.19e+02 ± 12.65
0.8	10	WiSARD	0.83 ± 0.00	3.40 ± 0.19	1.14e+01 ± 1.38
		Clus	0.83 ± 0.00	25.98 ± 1.15	9.71e+01 ± 2.99
		Mix	0.83 ± 0.01	19.55 ± 2.79	9.25e+01 ± 38.28
	20	WiSARD	0.83 ± 0.00	6.59 ± 0.38	2.95e+01 ± 7.22
		Clus	0.84 ± 0.00	40.17 ± 3.93	1.61e+02 ± 28.77
		Mix	0.84 ± 0.00	38.13 ± 1.04	1.69e+02 ± 44.98

Table B.42: Results for WiSARD Boosting Ensembles in Fashion MNIST Dataset with Local Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.81 ± 0.01	7.80 ± 0.18	24.77 ± 10.79	8.37e+01 ± 15.79
	Clus	0.81 ± 0.01	7.82 ± 0.15	25.44 ± 24.78	8.39e+01 ± 40.96
	Mix	0.81 ± 0.00	7.94 ± 0.30	23.54 ± 15.38	8.39e+01 ± 30.00
20	WiSARD	0.81 ± 0.00	8.09 ± 0.26	39.68 ± 8.37	1.55e+02 ± 10.03
	Clus	0.81 ± 0.00	8.39 ± 0.27	29.04 ± 14.25	1.35e+02 ± 36.94
	Mix	0.81 ± 0.00	8.11 ± 0.11	34.74 ± 2.60	1.41e+02 ± 10.74

Table B.43: Results for Borda Count Ensembles in Fashion MNIST Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.83 ± 0.00	2.97 ± 0.22	1.84e+01 ± 3.01
		Start at 1	0.82 ± 0.01	3.23 ± 0.08	2.19e+01 ± 10.24
		Dowdall	0.83 ± 0.01	3.27 ± 0.12	1.96e+01 ± 4.63
	20	Start at 0	0.83 ± 0.01	6.49 ± 0.06	4.94e+01 ± 16.35
		Start at 1	0.83 ± 0.00	6.24 ± 0.07	3.67e+01 ± 5.55
		Dowdall	0.83 ± 0.00	6.35 ± 0.08	4.72e+01 ± 6.18
0.8	10	Start at 0	0.83 ± 0.00	4.44 ± 0.07	2.46e+01 ± 5.10
		Start at 1	0.83 ± 0.00	4.37 ± 0.04	2.12e+01 ± 4.93
		Dowdall	0.83 ± 0.00	4.44 ± 0.07	2.09e+01 ± 4.94
	20	Start at 0	0.83 ± 0.00	8.47 ± 0.14	4.93e+01 ± 10.98
		Start at 1	0.84 ± 0.00	8.50 ± 0.07	4.47e+01 ± 7.78
		Dowdall	0.83 ± 0.00	8.52 ± 0.08	4.65e+01 ± 2.21

Table B.44: Results for Tie-break Ensembles in Fashion MNIST Dataset with Local Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.75 ± 0.01	3.08 ± 0.29	1.68e+01 ± 2.17
		Only Ties	0.74 ± 0.01	3.42 ± 0.15	1.79e+01 ± 1.39
		Threshold	0.74 ± 0.01	3.36 ± 0.03	1.91e+01 ± 2.53
	20	All Candidates	0.76 ± 0.01	6.31 ± 0.20	3.37e+01 ± 5.15
		Only Ties	0.76 ± 0.01	6.42 ± 0.07	3.66e+01 ± 3.32
		Threshold	0.76 ± 0.00	6.28 ± 0.05	3.44e+01 ± 2.16
0.8	10	All Candidates	0.74 ± 0.01	4.39 ± 0.08	1.74e+01 ± 0.73
		Only Ties	0.75 ± 0.00	4.37 ± 0.01	1.73e+01 ± 1.60
		Threshold	0.73 ± 0.02	4.35 ± 0.05	1.68e+01 ± 1.92
	20	All Candidates	0.78 ± 0.01	8.45 ± 0.01	3.40e+01 ± 2.06
		Only Ties	0.77 ± 0.01	8.53 ± 0.08	3.50e+01 ± 1.25
		Threshold	0.75 ± 0.00	8.39 ± 0.47	3.43e+01 ± 4.07

Table B.45: Results for Weighted Votes Ensembles in Fashion MNIST Dataset with Local Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.74 ± 0.01	3.37 ± 0.03	1.86e+01 ± 1.54
	20	0.77 ± 0.01	6.26 ± 0.07	3.27e+01 ± 2.99
0.8	10	0.75 ± 0.01	4.42 ± 0.05	1.71e+01 ± 0.43
	20	0.76 ± 0.02	8.65 ± 0.07	3.80e+01 ± 4.30

Table B.46: Results for WiSARD Bagging Ensembles in Fashion MNIST Dataset with Mean Threshold

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.82 ± 0.02	1.87 ± 0.07	1.31e+01 ± 4.27
		Clus	0.84 ± 0.00	13.32 ± 0.25	7.18e+01 ± 27.60
		Mix	0.84 ± 0.00	14.18 ± 1.63	1.00e+02 ± 42.25
	20	WiSARD	0.83 ± 0.01	3.83 ± 0.04	2.23e+01 ± 6.81
		Clus	0.84 ± 0.00	26.43 ± 2.10	1.53e+02 ± 32.80
		Mix	0.85 ± 0.00	26.31 ± 1.50	1.50e+02 ± 20.46
0.8	10	WiSARD	0.82 ± 0.01	2.76 ± 0.25	1.35e+01 ± 3.26
		Clus	0.83 ± 0.00	21.98 ± 7.33	9.16e+01 ± 26.75
		Mix	0.83 ± 0.00	15.98 ± 0.23	8.89e+01 ± 4.47
	20	WiSARD	0.84 ± 0.01	5.28 ± 0.15	2.13e+01 ± 2.25
		Clus	0.85 ± 0.00	33.59 ± 1.54	1.49e+02 ± 32.55
		Mix	0.84 ± 0.00	38.22 ± 3.13	1.74e+02 ± 34.29

Table B.47: Results for WiSARD Boosting Ensembles in Fashion MNIST Dataset with Mean Threshold

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.82 ± 0.00	8.13 ± 0.26	19.70 ± 8.28	7.50e+01 ± 12.60
	Clus	0.82 ± 0.00	8.19 ± 0.22	15.34 ± 2.45	6.72e+01 ± 7.13
	Mix	0.82 ± 0.00	8.37 ± 0.24	31.71 ± 13.43	9.95e+01 ± 23.69
20	WiSARD	0.82 ± 0.00	7.99 ± 0.04	35.66 ± 20.90	1.40e+02 ± 28.72
	Clus	0.82 ± 0.00	8.03 ± 0.07	32.28 ± 4.22	1.47e+02 ± 11.19
	Mix	0.82 ± 0.00	8.12 ± 0.33	43.14 ± 17.84	1.62e+02 ± 46.87

Table B.48: Results for Borda Count Ensembles in Fashion MNIST Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.82 ± 0.00	2.92 ± 0.07	2.75e+01 ± 3.67
		Start at 1	0.84 ± 0.00	2.71 ± 0.06	1.61e+01 ± 1.06
		Dowdall	0.83 ± 0.00	2.80 ± 0.10	2.02e+01 ± 5.19
	20	Start at 0	0.84 ± 0.01	5.38 ± 0.26	3.72e+01 ± 7.32
		Start at 1	0.83 ± 0.01	5.78 ± 0.16	5.31e+01 ± 11.73
		Dowdall	0.84 ± 0.00	5.37 ± 0.08	4.23e+01 ± 9.18
0.8	10	Start at 0	0.83 ± 0.00	3.82 ± 0.05	2.69e+01 ± 3.48
		Start at 1	0.83 ± 0.01	3.74 ± 0.13	2.42e+01 ± 5.06
		Dowdall	0.83 ± 0.01	3.67 ± 0.18	1.73e+01 ± 3.19
	20	Start at 0	0.83 ± 0.01	7.54 ± 0.10	5.26e+01 ± 7.27
		Start at 1	0.84 ± 0.00	7.33 ± 0.08	4.45e+01 ± 9.50
		Dowdall	0.84 ± 0.00	7.40 ± 0.05	4.56e+01 ± 8.01

Table B.49: Results for Tie-break Ensembles in Fashion MNIST Dataset with Mean Threshold

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.76 ± 0.01	2.95 ± 0.11	1.74e+01 ± 1.36
		Only Ties	0.76 ± 0.01	2.94 ± 0.15	1.71e+01 ± 2.08
		Threshold	0.76 ± 0.01	2.78 ± 0.04	1.57e+01 ± 0.29
	20	All Candidates	0.78 ± 0.00	5.52 ± 0.13	3.38e+01 ± 1.86
		Only Ties	0.78 ± 0.01	5.41 ± 0.07	3.23e+01 ± 1.10
		Threshold	0.77 ± 0.01	5.59 ± 0.17	3.53e+01 ± 1.90
0.8	10	All Candidates	0.77 ± 0.01	3.82 ± 0.07	1.59e+01 ± 1.13
		Only Ties	0.76 ± 0.01	3.81 ± 0.11	1.68e+01 ± 1.92
		Threshold	0.75 ± 0.01	3.91 ± 0.24	1.73e+01 ± 1.99
	20	All Candidates	0.77 ± 0.01	7.50 ± 0.10	3.50e+01 ± 2.29
		Only Ties	0.78 ± 0.01	7.39 ± 0.13	3.43e+01 ± 2.12
		Threshold	0.77 ± 0.01	7.52 ± 0.17	3.51e+01 ± 2.94

Table B.50: Results for Weighted Votes Ensembles in Fashion MNIST Dataset with Mean Threshold

Partition	wl	Acc	Training time	Test time
0.6	10	0.77 ± 0.01	2.86 ± 0.12	1.70e+01 ± 2.46
	20	0.78 ± 0.01	5.51 ± 0.10	3.40e+01 ± 1.98
0.8	10	0.77 ± 0.01	3.89 ± 0.12	1.77e+01 ± 1.07
	20	0.78 ± 0.00	7.30 ± 0.16	3.20e+01 ± 1.54

Table B.51: Results for WiSARD Bagging Ensembles in Fashion MNIST Dataset with Otsu's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.82 ± 0.01	1.54 ± 0.04	9.52e+00 ± 2.00
		Clus	0.83 ± 0.01	14.77 ± 1.32	1.02e+02 ± 42.94
		Mix	0.83 ± 0.00	13.71 ± 1.73	8.54e+01 ± 12.91
	20	WiSARD	0.83 ± 0.00	3.84 ± 0.22	2.34e+01 ± 3.93
		Clus	0.84 ± 0.00	26.82 ± 4.30	1.43e+02 ± 37.26
		Mix	0.84 ± 0.00	27.11 ± 4.76	1.44e+02 ± 28.33
0.8	10	WiSARD	0.82 ± 0.01	2.65 ± 0.09	1.58e+01 ± 6.16
		Clus	0.83 ± 0.00	22.56 ± 8.08	1.09e+02 ± 62.61
		Mix	0.83 ± 0.00	22.37 ± 4.65	9.09e+01 ± 35.02
	20	WiSARD	0.83 ± 0.00	5.13 ± 0.20	2.64e+01 ± 6.51
		Clus	0.84 ± 0.00	37.52 ± 1.45	2.04e+02 ± 50.23
		Mix	0.84 ± 0.00	38.13 ± 2.27	1.53e+02 ± 25.79

Table B.52: Results for WiSARD Boosting Ensembles in Fashion MNIST Dataset with Otsu's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.81 ± 0.01	8.18 ± 0.35	38.84 ± 26.91	1.05e+02 ± 46.34
	Clus	0.82 ± 0.00	7.99 ± 0.17	19.08 ± 15.50	6.99e+01 ± 25.64
	Mix	0.81 ± 0.00	8.16 ± 0.09	20.14 ± 2.42	8.11e+01 ± 4.72
20	WiSARD	0.81 ± 0.00	7.91 ± 0.09	27.54 ± 4.33	1.28e+02 ± 7.70
	Clus	0.82 ± 0.00	7.90 ± 0.07	33.83 ± 4.35	1.38e+02 ± 3.14
	Mix	0.82 ± 0.00	7.95 ± 0.17	29.47 ± 8.97	1.29e+02 ± 18.14

Table B.53: Results for Borda Count Ensembles in Fashion MNIST Dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.82 ± 0.00	2.68 ± 0.18	2.05e+01 ± 1.35
		Start at 1	0.82 ± 0.01	2.82 ± 0.13	2.19e+01 ± 6.09
		Dowdall	0.83 ± 0.01	2.75 ± 0.13	1.75e+01 ± 3.51
	20	Start at 0	0.83 ± 0.00	5.63 ± 0.13	4.07e+01 ± 3.65
		Start at 1	0.83 ± 0.01	5.53 ± 0.23	4.35e+01 ± 7.21
		Dowdall	0.83 ± 0.00	5.46 ± 0.09	4.33e+01 ± 6.38
0.8	10	Start at 0	0.82 ± 0.01	3.85 ± 0.13	2.37e+01 ± 8.79
		Start at 1	0.83 ± 0.00	3.72 ± 0.03	2.11e+01 ± 3.49
		Dowdall	0.82 ± 0.01	3.76 ± 0.08	2.51e+01 ± 1.86
	20	Start at 0	0.83 ± 0.00	7.29 ± 0.17	4.10e+01 ± 7.24
		Start at 1	0.83 ± 0.01	7.56 ± 0.37	4.99e+01 ± 12.30
		Dowdall	0.83 ± 0.00	7.48 ± 0.23	4.62e+01 ± 9.99

Table B.54: Results for Tie-break Ensembles in Fashion MNIST Dataset with Otsu's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.75 ± 0.01	2.76 ± 0.19	1.65e+01 ± 1.21
		Only Ties	0.75 ± 0.02	2.93 ± 0.11	1.67e+01 ± 1.03
		Threshold	0.75 ± 0.01	2.91 ± 0.14	1.69e+01 ± 1.75
	20	All Candidates	0.76 ± 0.01	5.70 ± 0.29	3.55e+01 ± 3.92
		Only Ties	0.77 ± 0.00	5.57 ± 0.07	3.39e+01 ± 0.86
		Threshold	0.76 ± 0.01	5.60 ± 0.14	3.53e+01 ± 2.45
0.8	10	All Candidates	0.75 ± 0.01	4.02 ± 0.09	1.78e+01 ± 0.32
		Only Ties	0.76 ± 0.01	3.96 ± 0.08	1.73e+01 ± 0.52
		Threshold	0.76 ± 0.01	3.84 ± 0.05	1.60e+01 ± 0.93
	20	All Candidates	0.77 ± 0.01	7.44 ± 0.10	3.35e+01 ± 2.05
		Only Ties	0.77 ± 0.01	7.38 ± 0.20	3.33e+01 ± 2.80
		Threshold	0.76 ± 0.00	7.29 ± 0.23	3.24e+01 ± 3.21

Table B.55: Results for Weighted Votes Ensembles in Fashion MNIST Dataset with Otsu's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.75 ± 0.01	2.98 ± 0.15	1.77e+01 ± 1.22
	20	0.76 ± 0.02	5.63 ± 0.26	3.47e+01 ± 3.46
0.8	10	0.76 ± 0.01	3.89 ± 0.16	1.64e+01 ± 1.16
	20	0.77 ± 0.01	7.75 ± 0.02	3.71e+01 ± 0.77

B.3.4 Yen's Binarization

The results are found in Tables B.56-B.60.

Table B.56: Results for WiSARD Bagging Ensembles in Fashion MNIST Dataset with Yen's Binarization

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.79 ± 0.00	2.20 ± 0.23	9.08e+00 ± 1.12
		Clus	0.78 ± 0.01	15.37 ± 3.31	8.62e+01 ± 48.92
		Mix	0.79 ± 0.01	16.07 ± 3.12	7.76e+01 ± 19.31
	20	WiSARD	0.80 ± 0.00	4.61 ± 0.11	2.43e+01 ± 2.54
		Clus	0.80 ± 0.00	30.51 ± 1.83	1.67e+02 ± 22.26
		Mix	0.80 ± 0.00	25.11 ± 4.80	1.20e+02 ± 6.03
0.8	10	WiSARD	0.79 ± 0.00	3.22 ± 0.17	1.28e+01 ± 3.42
		Clus	0.79 ± 0.00	19.34 ± 1.94	6.73e+01 ± 21.75
		Mix	0.79 ± 0.00	18.25 ± 1.95	8.73e+01 ± 41.77
	20	WiSARD	0.80 ± 0.00	6.51 ± 0.11	2.52e+01 ± 4.00
		Clus	0.80 ± 0.00	35.76 ± 2.60	1.35e+02 ± 31.23
		Mix	0.80 ± 0.00	39.44 ± 7.04	1.79e+02 ± 42.21

Table B.57: Results for WiSARD Boosting ensembles in Fashion MNIST dataset with Yen's Binarization

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.77 ± 0.00	8.21 ± 0.14	42.13 ± 23.07	1.08e+02 ± 39.82
	Clus	0.77 ± 0.00	7.45 ± 0.20	6.90 ± 1.97	4.53e+01 ± 7.84
	Mix	0.76 ± 0.01	7.71 ± 0.26	24.13 ± 19.98	7.78e+01 ± 32.41
20	WiSARD	0.77 ± 0.00	7.71 ± 0.10	20.22 ± 1.86	1.11e+02 ± 7.79
	Clus	0.77 ± 0.00	7.90 ± 0.09	27.82 ± 6.08	1.28e+02 ± 12.30
	Mix	0.77 ± 0.00	7.98 ± 0.28	21.47 ± 1.40	1.17e+02 ± 3.65

Table B.58: Results for Borda Count Ensembles in Fashion MNIST Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.79 ± 0.01	3.23 ± 0.05	2.17e+01 ± 5.70
		Start at 1	0.78 ± 0.01	3.19 ± 0.05	2.43e+01 ± 8.48
		Dowdall	0.79 ± 0.00	3.19 ± 0.04	2.09e+01 ± 3.31
	20	Start at 0	0.80 ± 0.00	6.20 ± 0.04	4.43e+01 ± 9.06
		Start at 1	0.80 ± 0.00	6.17 ± 0.08	4.20e+01 ± 5.85
		Dowdall	0.80 ± 0.01	6.17 ± 0.03	4.01e+01 ± 7.39
0.8	10	Start at 0	0.79 ± 0.01	4.30 ± 0.05	2.25e+01 ± 2.98
		Start at 1	0.79 ± 0.00	4.28 ± 0.05	2.11e+01 ± 5.63
		Dowdall	0.79 ± 0.01	4.25 ± 0.05	2.10e+01 ± 5.88
	20	Start at 0	0.80 ± 0.00	8.47 ± 0.14	4.51e+01 ± 2.18
		Start at 1	0.80 ± 0.00	8.26 ± 0.01	4.58e+01 ± 6.10
		Dowdall	0.80 ± 0.00	8.11 ± 0.38	4.87e+01 ± 4.10

Table B.59: Results for Tie-break Ensembles in Fashion MNIST Dataset with Yen's Binarization

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.64 ± 0.02	3.29 ± 0.01	1.64e+01 ± 0.76
		Only Ties	0.62 ± 0.04	3.34 ± 0.06	1.79e+01 ± 1.83
		Threshold	0.61 ± 0.01	3.26 ± 0.04	1.72e+01 ± 1.67
	20	All Candidates	0.68 ± 0.02	6.23 ± 0.03	3.65e+01 ± 2.93
		Only Ties	0.69 ± 0.00	6.14 ± 0.08	3.23e+01 ± 2.60
		Threshold	0.66 ± 0.01	6.24 ± 0.07	3.53e+01 ± 2.09
0.8	10	All Candidates	0.65 ± 0.01	4.44 ± 0.10	1.74e+01 ± 1.69
		Only Ties	0.65 ± 0.01	4.39 ± 0.06	1.73e+01 ± 0.74
		Threshold	0.62 ± 0.01	4.27 ± 0.09	1.83e+01 ± 0.22
	20	All Candidates	0.68 ± 0.01	8.48 ± 0.02	3.63e+01 ± 2.64
		Only Ties	0.70 ± 0.01	8.31 ± 0.04	3.32e+01 ± 1.64
		Threshold	0.64 ± 0.01	8.27 ± 0.37	3.45e+01 ± 2.80

B.4 IMDb Dataset

The results of IMDb dataset are found in Tables B.61-B.65.

Table B.60: Results for Weighted Votes Ensembles in Fashion MNIST Dataset with Yen's Binarization

Partition	wl	Acc	Training time	Test time
0.6	10	0.67 ± 0.00	3.31 ± 0.01	1.71e+01 ± 1.05
	20	0.70 ± 0.01	6.21 ± 0.10	3.36e+01 ± 1.08
0.8	10	0.67 ± 0.00	4.39 ± 0.04	1.83e+01 ± 1.02
	20	0.69 ± 0.00	8.44 ± 0.01	3.64e+01 ± 0.98

Table B.61: Results for WiSARD Bagging Ensembles in IMDb Dataset

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.66 ± 0.00	4.59 ± 0.26	2.26e+01 ± 1.96
		Clus	0.66 ± 0.01	30.39 ± 5.48	1.66e+02 ± 61.45
		Mix	0.66 ± 0.00	26.30 ± 3.38	1.31e+02 ± 31.58
	20	WiSARD	0.70 ± 0.01	8.35 ± 0.40	3.98e+01 ± 3.44
		Clus	0.70 ± 0.01	53.97 ± 3.78	2.90e+02 ± 14.90
		Mix	0.70 ± 0.01	53.10 ± 4.63	2.70e+02 ± 55.29
0.8	10	WiSARD	0.66 ± 0.01	5.71 ± 0.69	2.08e+01 ± 3.53
		Clus	0.67 ± 0.00	34.52 ± 1.09	1.30e+02 ± 19.10
		Mix	0.66 ± 0.00	37.67 ± 3.04	1.69e+02 ± 23.19
	20	WiSARD	0.70 ± 0.00	11.76 ± 0.45	4.30e+01 ± 2.14
		Clus	0.70 ± 0.01	71.19 ± 2.66	2.91e+02 ± 38.87
		Mix	0.70 ± 0.01	80.93 ± 6.23	3.18e+02 ± 50.51

Table B.62: Results for WiSARD Boosting Ensembles in IMDb Dataset

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.70 ± 0.01	3.43 ± 0.64	7.45 ± 3.04	6.40e+02 ± 59.47
	Clus	0.70 ± 0.01	3.67 ± 0.90	9.26 ± 6.21	6.51e+02 ± 75.67
	Mix	0.70 ± 0.00	2.89 ± 0.32	5.92 ± 1.73	6.03e+02 ± 25.83
20	WiSARD	0.78 ± 0.01	3.10 ± 0.13	14.61 ± 1.13	1.28e+03 ± 23.34
	Clus	0.78 ± 0.01	3.41 ± 0.29	18.36 ± 3.61	1.32e+03 ± 46.63
	Mix	0.78 ± 0.00	3.74 ± 0.37	19.79 ± 5.50	1.36e+03 ± 61.79

B.5 MNIST Dataset

The results of MNIST dataset are found in Tables B.66-B.70.

B.6 MovieLens Dataset

The results of MovieLens dataset are found in Tables B.71-B.75.

Table B.63: Results for Borda Count Ensembles in IMDb Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.66 ± 0.00	9.02 ± 0.68	9.86e+01 ± 8.25
		Start at 1	0.66 ± 0.00	9.73 ± 1.08	9.29e+01 ± 5.04
		Dowdall	0.66 ± 0.01	9.44 ± 0.57	9.25e+01 ± 2.43
	20	Start at 0	0.70 ± 0.01	18.15 ± 1.24	1.25e+02 ± 7.51
		Start at 1	0.70 ± 0.01	17.15 ± 1.41	1.20e+02 ± 8.11
		Dowdall	0.70 ± 0.00	17.48 ± 1.36	1.22e+02 ± 5.56
0.8	10	Start at 0	0.66 ± 0.00	11.61 ± 0.26	8.93e+01 ± 0.93
		Start at 1	0.66 ± 0.00	11.28 ± 0.52	8.06e+01 ± 2.21
		Dowdall	0.66 ± 0.00	11.17 ± 0.57	7.93e+01 ± 1.97
	20	Start at 0	0.70 ± 0.00	23.24 ± 1.10	1.21e+02 ± 4.64
		Start at 1	0.70 ± 0.01	22.80 ± 1.06	1.19e+02 ± 5.80
		Dowdall	0.69 ± 0.01	24.11 ± 1.10	1.26e+02 ± 4.37

Table B.64: Results for Tie-break Ensembles in IMDb Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.66 ± 0.03	9.23 ± 0.24	1.00e+02 ± 3.34
		Only Ties	0.64 ± 0.04	9.38 ± 0.66	9.77e+01 ± 1.70
		Threshold	0.66 ± 0.02	9.01 ± 0.12	9.54e+01 ± 1.47
	20	All Candidates	0.60 ± 0.05	16.78 ± 0.89	1.25e+02 ± 5.36
		Only Ties	0.64 ± 0.07	17.27 ± 0.63	1.27e+02 ± 1.66
		Threshold	0.65 ± 0.07	17.87 ± 0.68	1.32e+02 ± 6.38
0.8	10	All Candidates	0.65 ± 0.03	12.77 ± 1.88	8.51e+01 ± 5.72
		Only Ties	0.63 ± 0.07	11.77 ± 0.68	8.10e+01 ± 2.51
		Threshold	0.62 ± 0.02	11.56 ± 1.27	8.24e+01 ± 2.99
	20	All Candidates	0.63 ± 0.07	23.87 ± 1.62	1.32e+02 ± 8.30
		Only Ties	0.66 ± 0.03	23.48 ± 0.89	1.29e+02 ± 3.17
		Threshold	0.65 ± 0.03	23.43 ± 1.68	1.30e+02 ± 4.36

Table B.65: Results for Weighted Votes Ensembles in IMDb Dataset

Partition	wl	Acc	Training time	Test time
0.6	10	0.64 ± 0.05	9.10 ± 0.74	9.62e+01 ± 4.42
	20	0.65 ± 0.03	17.10 ± 0.56	1.25e+02 ± 3.99
0.8	10	0.63 ± 0.06	11.98 ± 0.22	8.28e+01 ± 0.34
	20	0.66 ± 0.02	23.34 ± 1.14	1.28e+02 ± 4.53

Table B.66: Results for WiSARD Bagging Ensembles in MNIST Dataset

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.92 ± 0.00	1.59 ± 0.05	8.38e+00 ± 1.09
		Clus	0.92 ± 0.00	11.09 ± 1.78	8.82e+01 ± 9.85
		Mix	0.92 ± 0.01	11.38 ± 0.92	1.08e+02 ± 34.72
	20	WiSARD	0.93 ± 0.00	3.20 ± 0.10	1.91e+01 ± 2.41
		Clus	0.93 ± 0.00	24.90 ± 2.19	2.31e+02 ± 30.14
		Mix	0.93 ± 0.00	20.86 ± 0.87	1.86e+02 ± 63.84
0.8	10	WiSARD	0.92 ± 0.01	2.13 ± 0.03	1.03e+01 ± 2.37
		Clus	0.92 ± 0.01	14.92 ± 2.56	1.21e+02 ± 29.78
		Mix	0.93 ± 0.00	16.07 ± 1.39	1.32e+02 ± 51.28
	20	WiSARD	0.93 ± 0.00	4.28 ± 0.07	1.94e+01 ± 2.79
		Clus	0.93 ± 0.00	32.69 ± 3.50	2.30e+02 ± 63.53
		Mix	0.93 ± 0.00	24.28 ± 1.94	1.92e+02 ± 51.40

Table B.67: Results for WiSARD Boosting Ensembles in MNIST Dataset

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.91 ± 0.00	8.14 ± 0.20	34.19 ± 23.38	9.95e+01 ± 40.95
	Clus	0.91 ± 0.01	7.58 ± 0.41	22.73 ± 24.69	7.14e+01 ± 47.09
	Mix	0.91 ± 0.01	7.78 ± 0.09	17.02 ± 1.81	6.88e+01 ± 5.50
20	WiSARD	0.92 ± 0.00	7.73 ± 0.05	29.30 ± 5.61	1.24e+02 ± 11.66
	Clus	0.91 ± 0.00	7.89 ± 0.19	34.53 ± 16.59	1.33e+02 ± 35.15
	Mix	0.91 ± 0.00	7.90 ± 0.22	23.24 ± 6.31	1.14e+02 ± 13.66

Table B.68: Results for Borda Count Ensembles in MNIST Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.92 ± 0.01	2.36 ± 0.11	2.02e+01 ± 4.48
		Start at 1	0.92 ± 0.00	2.56 ± 0.02	1.95e+01 ± 1.12
		Dowdall	0.92 ± 0.01	2.34 ± 0.09	1.63e+01 ± 1.47
	20	Start at 0	0.92 ± 0.01	4.91 ± 0.16	4.48e+01 ± 3.77
		Start at 1	0.93 ± 0.00	4.82 ± 0.02	3.87e+01 ± 2.32
		Dowdall	0.92 ± 0.01	4.97 ± 0.13	4.51e+01 ± 9.09
0.8	10	Start at 0	0.92 ± 0.01	3.22 ± 0.09	1.87e+01 ± 2.77
		Start at 1	0.92 ± 0.01	3.14 ± 0.04	1.68e+01 ± 4.27
		Dowdall	0.92 ± 0.01	3.28 ± 0.08	2.25e+01 ± 4.91
	20	Start at 0	0.93 ± 0.00	6.45 ± 0.16	3.75e+01 ± 9.39
		Start at 1	0.93 ± 0.00	6.50 ± 0.17	4.11e+01 ± 5.86
		Dowdall	0.93 ± 0.00	6.59 ± 0.10	4.20e+01 ± 2.21

Table B.69: Results for Tie-break Ensembles in MNIST Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.60 ± 0.05	2.38 ± 0.08	1.30e+01 ± 0.50
		Only Ties	0.65 ± 0.03	2.50 ± 0.05	1.39e+01 ± 0.39
		Threshold	0.55 ± 0.04	2.49 ± 0.07	1.41e+01 ± 0.38
	20	All Candidates	0.64 ± 0.07	5.11 ± 0.11	3.04e+01 ± 1.22
		Only Ties	0.73 ± 0.01	4.83 ± 0.04	2.97e+01 ± 0.76
		Threshold	0.57 ± 0.04	5.00 ± 0.17	3.08e+01 ± 0.84
0.8	10	All Candidates	0.65 ± 0.04	3.31 ± 0.03	1.42e+01 ± 0.33
		Only Ties	0.66 ± 0.01	3.31 ± 0.01	1.38e+01 ± 0.36
		Threshold	0.57 ± 0.02	3.26 ± 0.07	1.44e+01 ± 0.23
	20	All Candidates	0.70 ± 0.07	6.70 ± 0.18	3.17e+01 ± 1.03
		Only Ties	0.70 ± 0.07	6.67 ± 0.14	3.02e+01 ± 0.93
		Threshold	0.58 ± 0.03	6.67 ± 0.14	3.05e+01 ± 0.64

Table B.70: Results for Weighted Votes Ensembles in MNIST Dataset

Partition	wl	Acc	Training time	Test time
0.6	10	0.75 ± 0.02	2.60 ± 0.09	1.41e+01 ± 0.19
	20	0.80 ± 0.01	5.16 ± 0.04	3.08e+01 ± 0.45
0.8	10	0.74 ± 0.03	3.24 ± 0.02	1.43e+01 ± 0.58
	20	0.82 ± 0.01	6.64 ± 0.13	3.02e+01 ± 0.61

Table B.71: Results for WiSARD Bagging Ensembles in MovieLens Dataset

Partition	wl	Models	Acc	Training time	Test time
0.6	10	WiSARD	0.70 ± 0.01	0.17 ± 0.00	1.55e-01 ± 0.01
		Clus	0.71 ± 0.00	0.96 ± 0.06	9.63e-01 ± 0.13
		Mix	0.70 ± 0.00	0.94 ± 0.10	9.32e-01 ± 0.17
	20	WiSARD	0.70 ± 0.00	0.34 ± 0.01	3.19e-01 ± 0.01
		Clus	0.70 ± 0.01	1.83 ± 0.21	1.79e+00 ± 0.34
		Mix	0.70 ± 0.00	1.95 ± 0.01	2.12e+00 ± 0.12
0.8	10	WiSARD	0.69 ± 0.00	0.24 ± 0.01	1.77e-01 ± 0.01
		Clus	0.70 ± 0.00	1.24 ± 0.14	9.83e-01 ± 0.32
		Mix	0.70 ± 0.00	1.24 ± 0.00	8.93e-01 ± 0.07
	20	WiSARD	0.69 ± 0.01	0.47 ± 0.01	3.50e-01 ± 0.01
		Clus	0.70 ± 0.00	2.58 ± 0.33	1.90e+00 ± 0.31
		Mix	0.70 ± 0.00	2.67 ± 0.39	2.24e+00 ± 0.51

Table B.72: Results for WiSARD Boosting Ensembles in MovieLens Dataset

wl	Models	Acc	Training time	Validation time	Test Time
10	WiSARD	0.70 ± 0.00	0.07 ± 0.01	0.11 ± 0.01	4.70e+00 ± 0.06
	Clus	0.70 ± 0.00	0.08 ± 0.01	0.13 ± 0.01	4.78e+00 ± 0.05
	Mix	0.70 ± 0.00	0.08 ± 0.01	0.12 ± 0.03	4.79e+00 ± 0.11
20	WiSARD	0.70 ± 0.00	0.08 ± 0.01	0.20 ± 0.01	8.96e+00 ± 0.06
	Clus	0.70 ± 0.00	0.07 ± 0.01	0.16 ± 0.02	8.74e+00 ± 0.08
	Mix	0.70 ± 0.00	0.08 ± 0.00	0.19 ± 0.00	8.86e+00 ± 0.02

Table B.73: Results for Borda Count Ensembles in MovieLens Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	Start at 0	0.69 ± 0.00	0.31 ± 0.02	8.35e-01 ± 0.04
		Start at 1	0.70 ± 0.01	0.32 ± 0.04	8.59e-01 ± 0.05
		Dowdall	0.70 ± 0.01	0.33 ± 0.01	8.47e-01 ± 0.02
	20	Start at 0	0.69 ± 0.01	0.66 ± 0.01	1.30e+00 ± 0.01
		Start at 1	0.70 ± 0.00	0.64 ± 0.04	1.25e+00 ± 0.05
		Dowdall	0.70 ± 0.00	0.64 ± 0.05	1.29e+00 ± 0.08
0.8	10	Start at 0	0.69 ± 0.01	0.42 ± 0.02	8.71e-01 ± 0.02
		Start at 1	0.69 ± 0.01	0.45 ± 0.03	9.07e-01 ± 0.04
		Dowdall	0.70 ± 0.00	0.40 ± 0.01	8.46e-01 ± 0.01
	20	Start at 0	0.70 ± 0.01	0.84 ± 0.04	1.29e+00 ± 0.06
		Start at 1	0.70 ± 0.00	0.83 ± 0.03	1.28e+00 ± 0.03
		Dowdall	0.70 ± 0.00	0.84 ± 0.04	1.29e+00 ± 0.08

Table B.74: Results for Tie-break Ensembles in MovieLens Dataset

Partition	wl	Policy	Acc	Training time	Test time
0.6	10	All Candidates	0.46 ± 0.03	0.30 ± 0.03	6.99e-01 ± 0.03
		Only Ties	0.45 ± 0.04	0.31 ± 0.01	7.15e-01 ± 0.03
		Threshold	0.42 ± 0.02	0.30 ± 0.02	6.97e-01 ± 0.01
	20	All Candidates	0.45 ± 0.03	0.60 ± 0.02	1.07e+00 ± 0.03
		Only Ties	0.43 ± 0.03	0.62 ± 0.02	1.10e+00 ± 0.02
		Threshold	0.39 ± 0.03	0.63 ± 0.02	1.10e+00 ± 0.02
0.8	10	All Candidates	0.44 ± 0.01	0.41 ± 0.01	7.18e-01 ± 0.01
		Only Ties	0.40 ± 0.03	0.41 ± 0.04	7.25e-01 ± 0.02
		Threshold	0.47 ± 0.04	0.40 ± 0.02	7.40e-01 ± 0.06
	20	All Candidates	0.42 ± 0.02	0.81 ± 0.02	1.10e+00 ± 0.01
		Only Ties	0.43 ± 0.04	0.81 ± 0.04	1.10e+00 ± 0.03
		Threshold	0.44 ± 0.03	0.82 ± 0.02	1.11e+00 ± 0.03

Table B.75: Results for Weighted Votes Ensembles in MovieLens Dataset

Partition	wl	Acc	Training time	Test time
0.6	10	0.42 ± 0.02	0.30 ± 0.01	6.96e-01 ± 0.01
	20	0.44 ± 0.05	0.62 ± 0.03	1.08e+00 ± 0.03
0.8	10	0.42 ± 0.01	0.39 ± 0.02	7.08e-01 ± 0.02
	20	0.41 ± 0.05	0.78 ± 0.02	1.08e+00 ± 0.03

Appendix C

Comparison between WiSARD and ClusWiSARD in Ensemble Experiments

Here are displayed the comparison between WiSARD and ClusWiSARD in the datasets presented in Chapter 4:

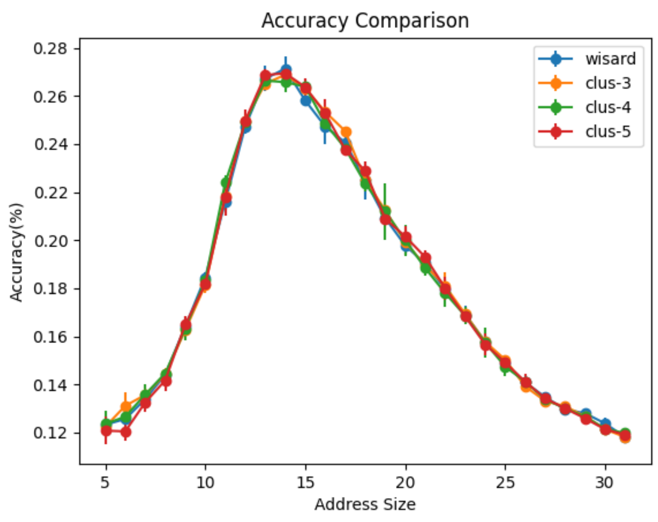


Figure C.1: Comparison of accuracy between WiSARD and ClusWiSARD in Cifar10 dataset with local threshold.

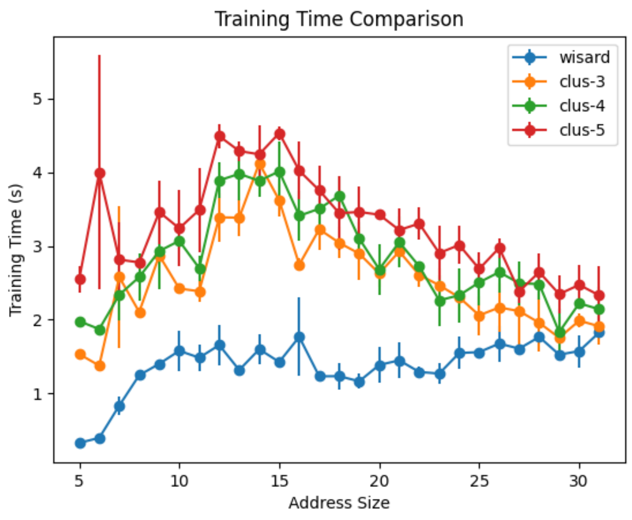


Figure C.2: Comparison of training time between WiSARD and ClusWiSARD in Cifar10 dataset with local threshold.

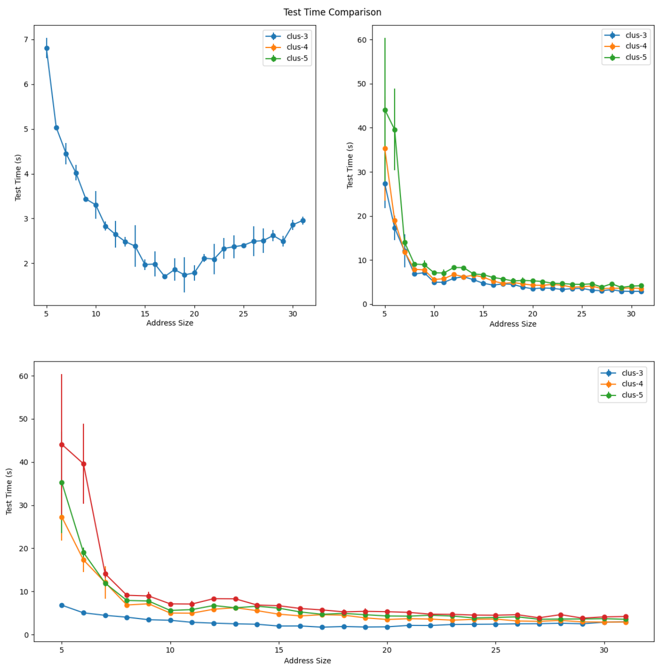


Figure C.3: Comparison of test time between WiSARD and ClusWiSARD in Cifar10 dataset with local threshold.

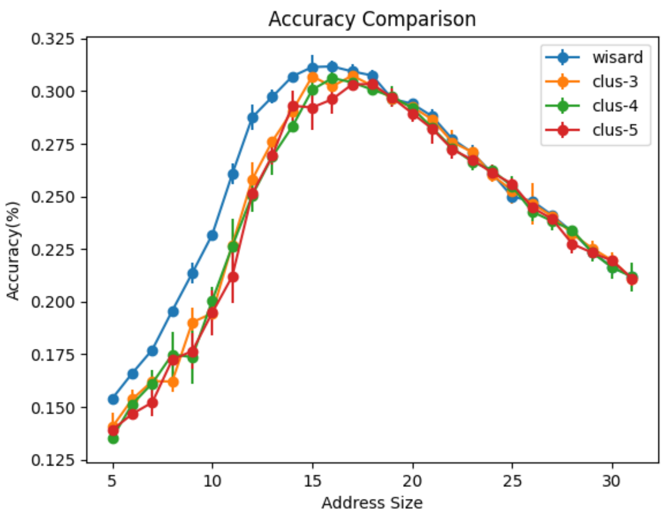


Figure C.4: Comparison of accuracy between WiSARD and ClusWiSARD in Cifar10 dataset with mean threshold.

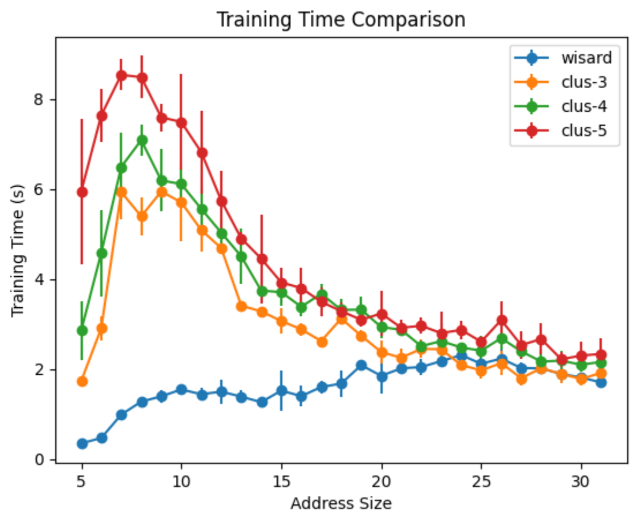


Figure C.5: Comparison of training time between WiSARD and ClusWiSARD in Cifar10 dataset with mean threshold.

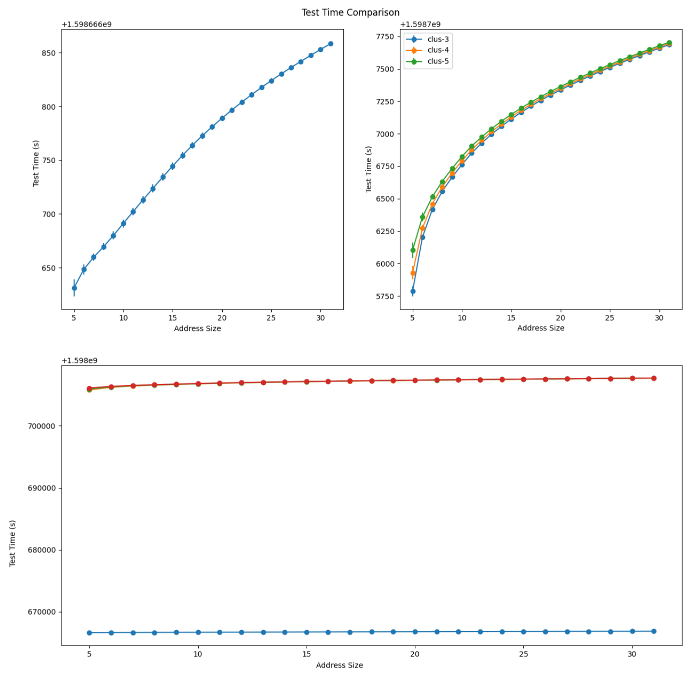


Figure C.6: Comparison of test time between WiSARD and ClusWiSARD in Cifar10 dataset with mean threshold.

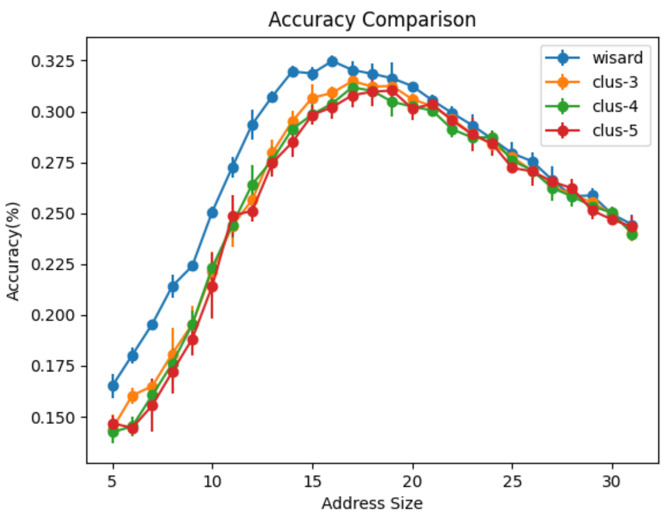


Figure C.7: Comparison of accuracy between WiSARD and ClusWiSARD in Cifar10 dataset with Otsu's Binarization

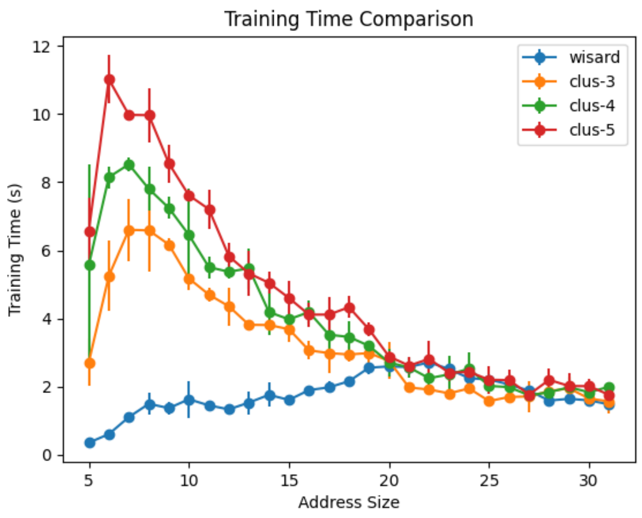


Figure C.8: Comparison of training time between WiSARD and ClusWiSARD in Cifar10 dataset with Otsu's Binarization.

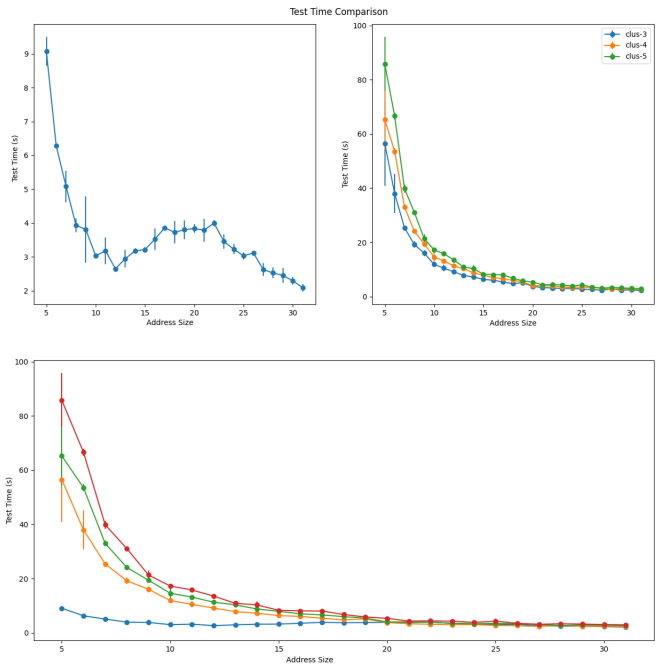


Figure C.9: Comparison of test time between WiSARD and ClusWiSARD in Cifar10 dataset with Otsu's Binarization.

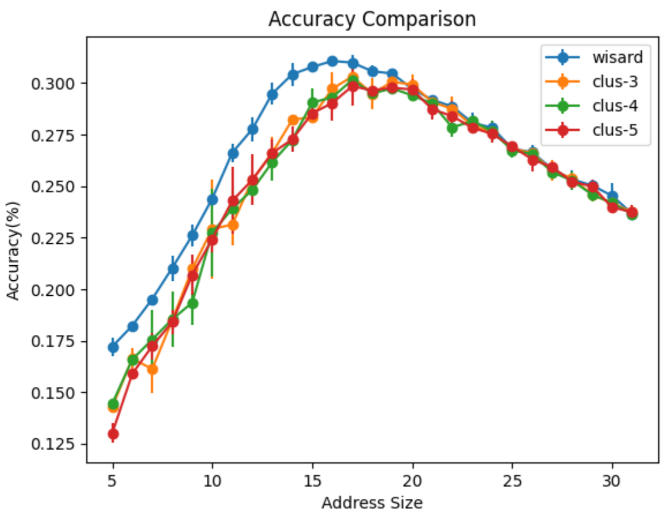


Figure C.10: Comparison of accuracy between WiSARD and ClusWiSARD in Cifar10 dataset with Yen's Binarization.

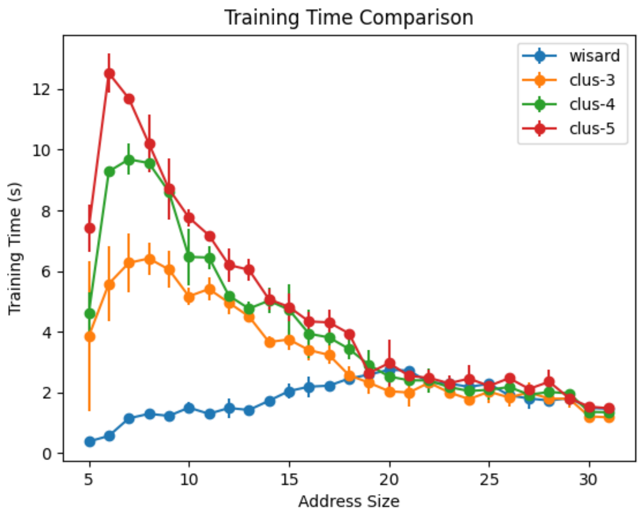


Figure C.11: Comparison of training time between WiSARD and ClusWiSARD in Cifar10 dataset with Yen's Binarization.

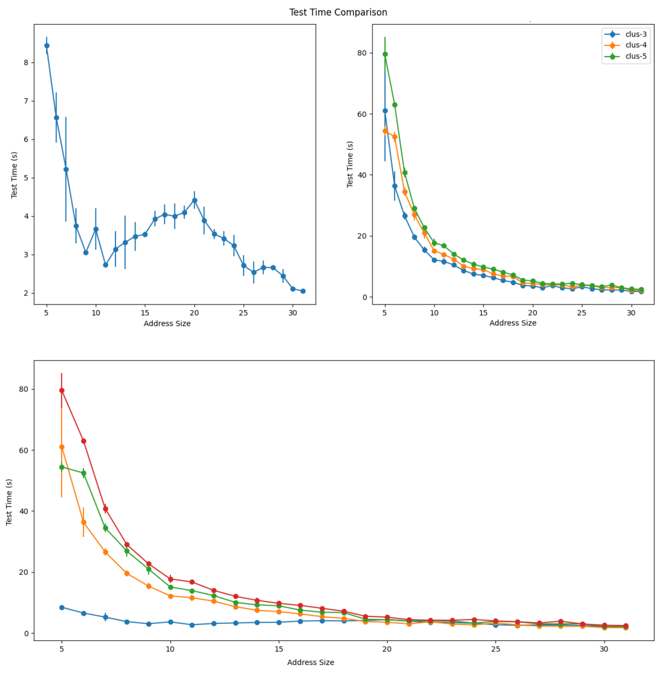


Figure C.12: Comparison of test time between WiSARD and ClusWiSARD in Cifar10 dataset with Yen's Binarization.

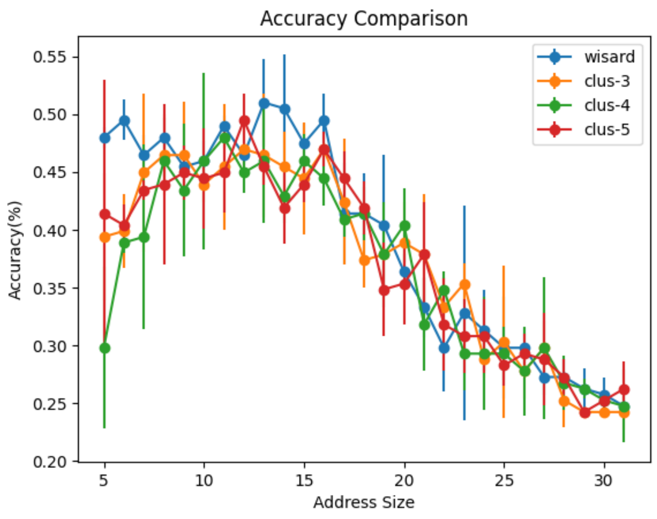


Figure C.13: Comparison of accuracy between WiSARD and ClusWiSARD in CKP dataset with local threshold.

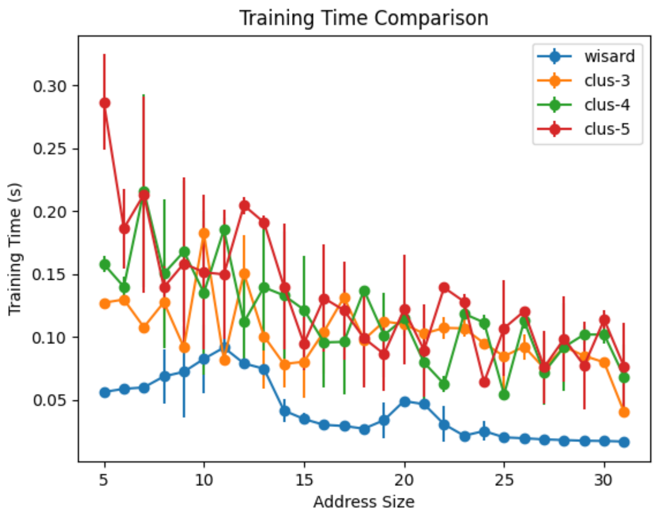


Figure C.14: Comparison of training time between WiSARD and ClusWiSARD in CKP with local threshold.

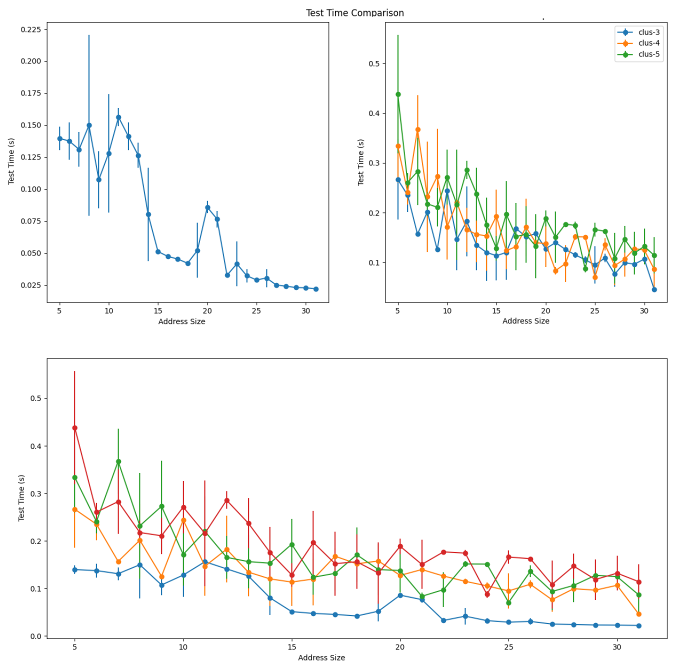


Figure C.15: Comparison of test time between WiSARD and ClusWiSARD in CKP dataset with local threshold.

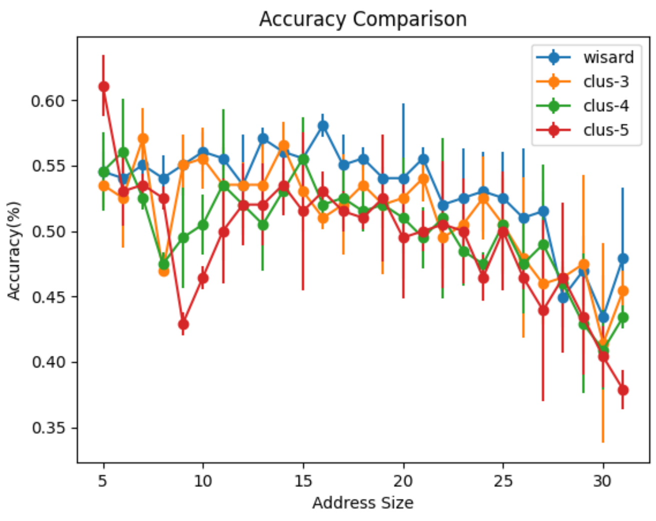


Figure C.16: Comparison of accuracy between WiSARD and ClusWiSARD in CKP dataset with mean threshold.

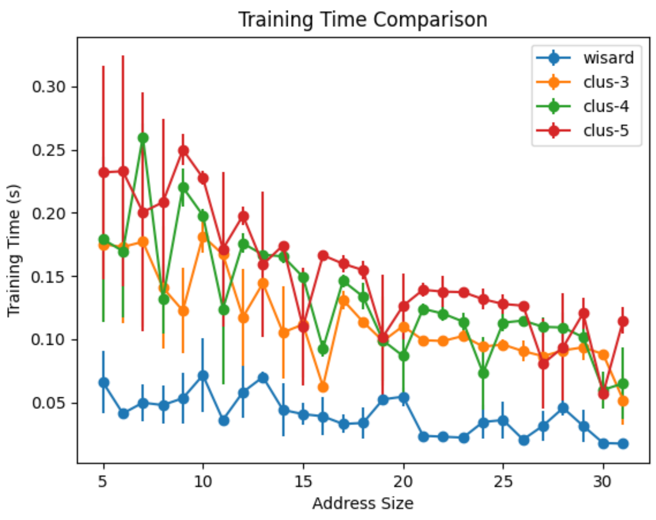


Figure C.17: Comparison of training time between WiSARD and ClusWiSARD in CKP with mean threshold.

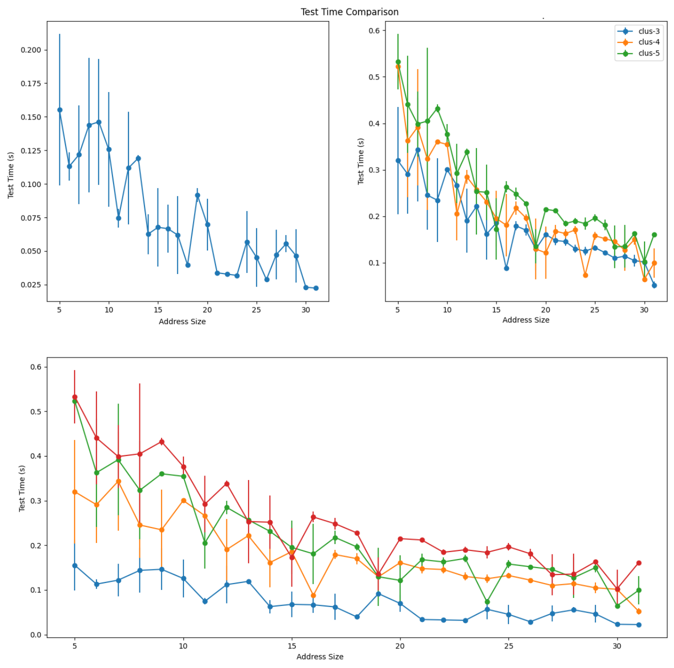


Figure C.18: Comparison of test time between WiSARD and ClusWiSARD in CKP dataset with mean threshold.

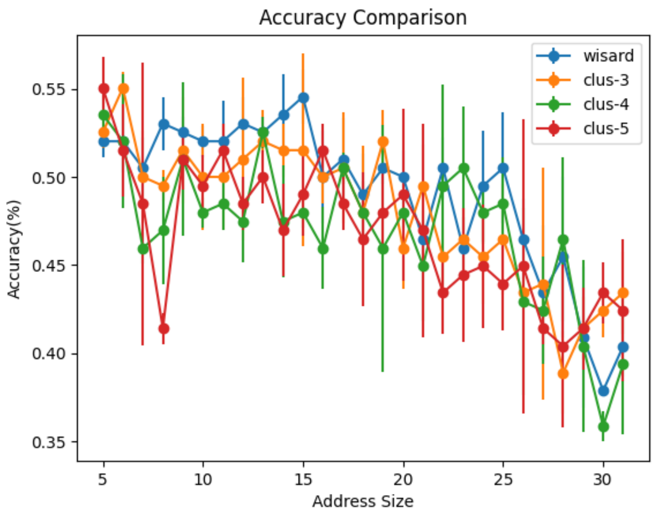


Figure C.19: Comparison of accuracy between WiSARD and ClusWiSARD in CKP dataset with Otsu's Binarization.

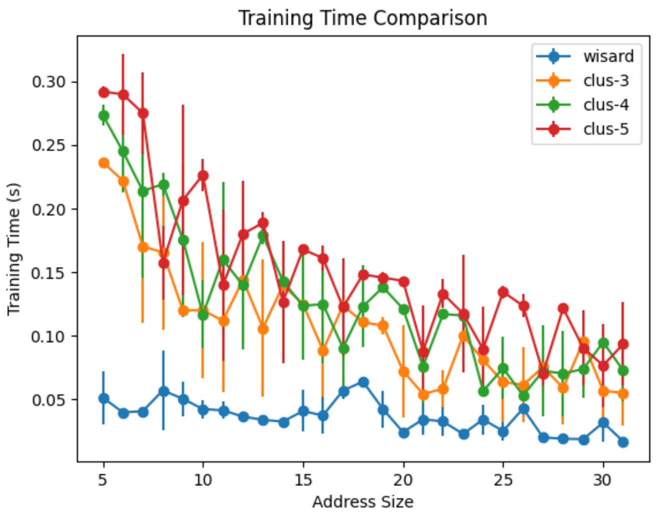


Figure C.20: Comparison of training time between WiSARD and ClusWiSARD in CKP with Otsu's Binarization.

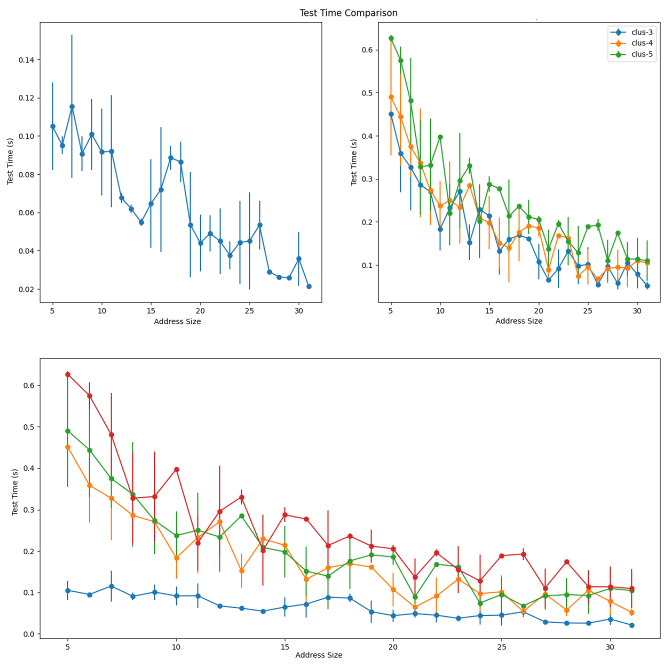


Figure C.21: Comparison of test time between WiSARD and ClusWiSARD in CKP dataset with Otsu's Binarization.

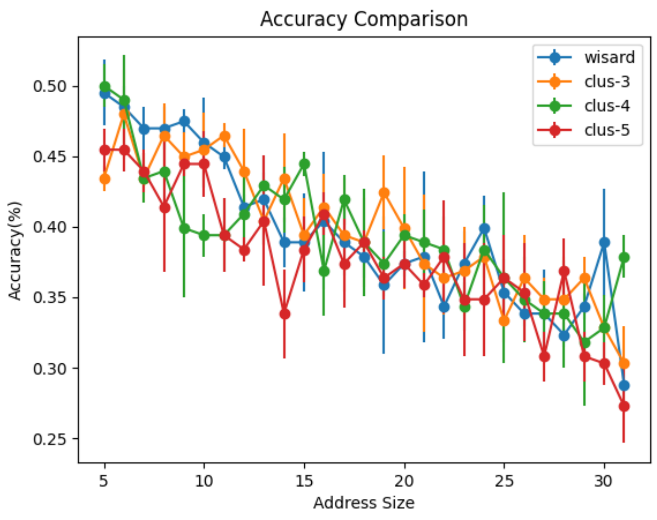


Figure C.22: Comparison of accuracy between WiSARD and ClusWiSARD in CKP dataset with Yen's Binarization.

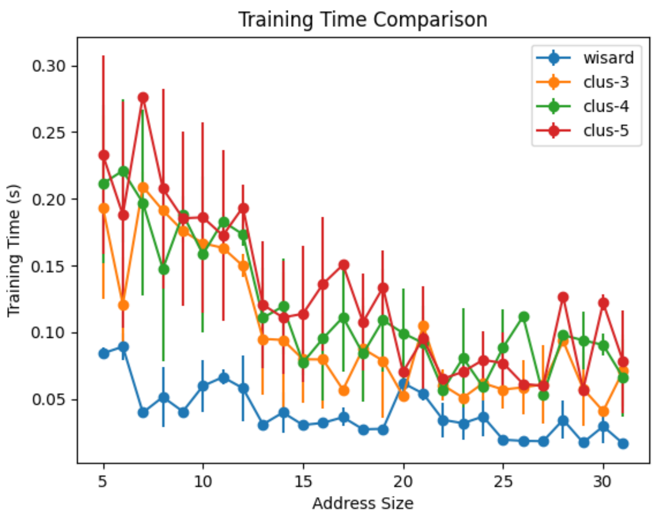


Figure C.23: Comparison of training time between WiSARD and ClusWiSARD in CKP with Yen's Binarization.

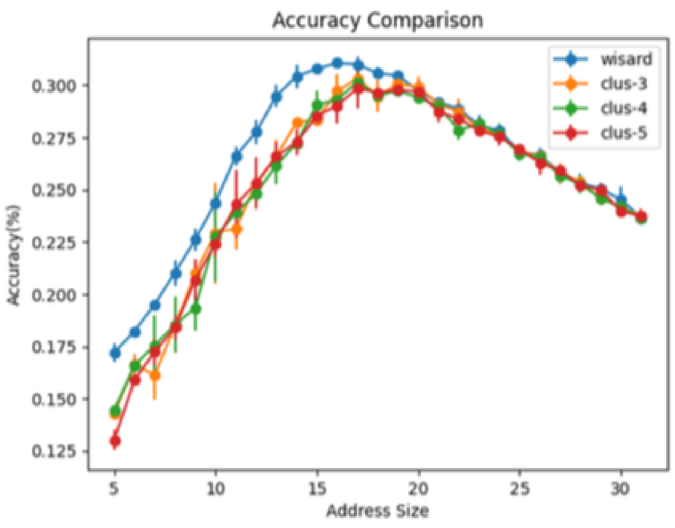


Figure C.24: Comparison of test time between WiSARD and ClusWiSARD in CKP dataset with Yen's Binarization.

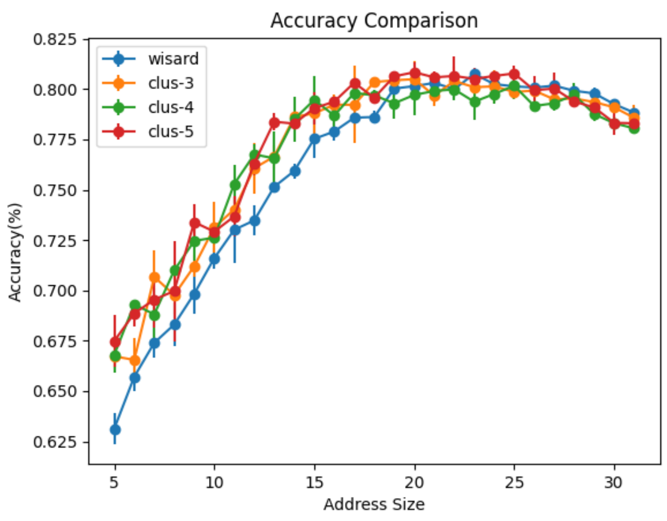


Figure C.25: Comparison of accuracy between WiSARD and ClusWiSARD in Fashion MNIST dataset with local threshold.

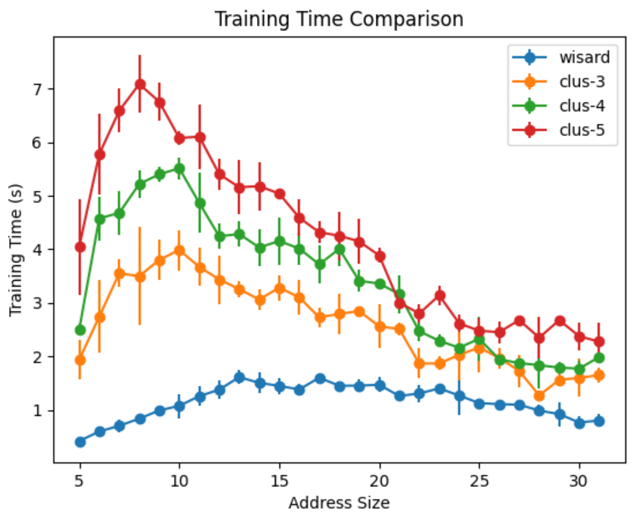


Figure C.26: Comparison of training time between WiSARD and ClusWiSARD in Fashion MNIST dataset with local threshold.

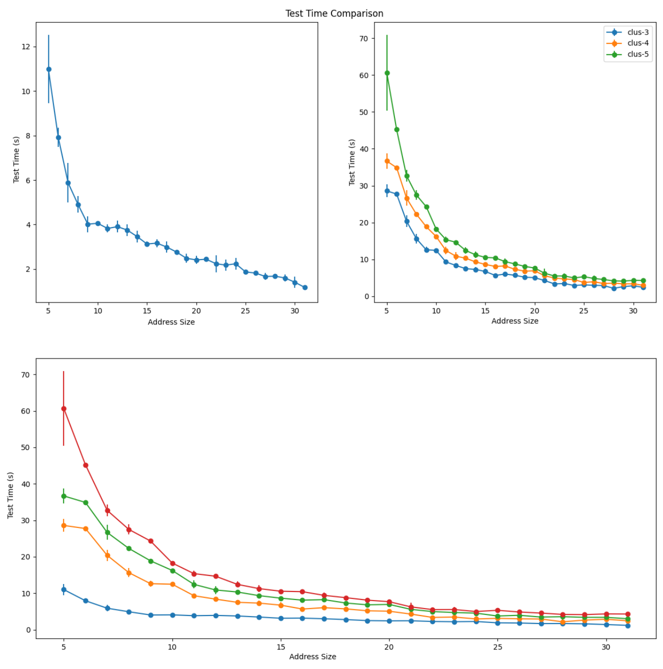


Figure C.27: Comparison of test time between WiSARD and ClusWiSARD in Fashion MNIST dataset with local threshold.

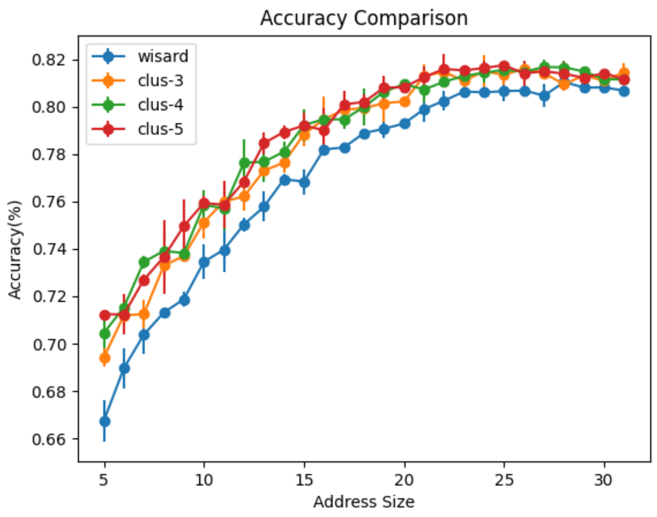


Figure C.28: Comparison of accuracy between WiSARD and ClusWiSARD in Fashion MNIST dataset with mean threshold.

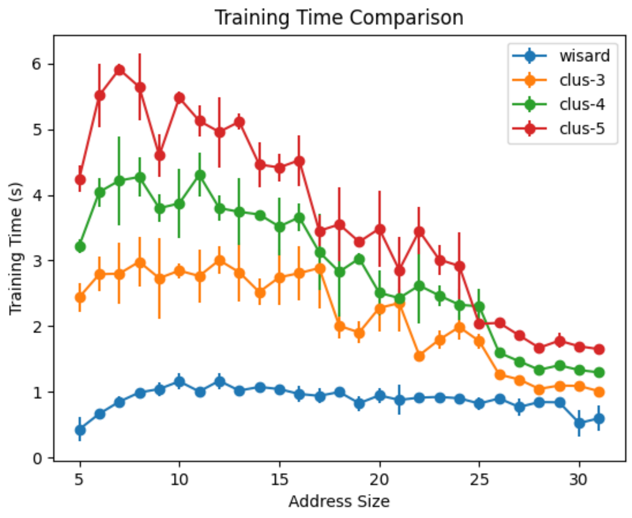


Figure C.29: Comparison of training time between WiSARD and ClusWiSARD in Fashion MNIST with mean threshold.

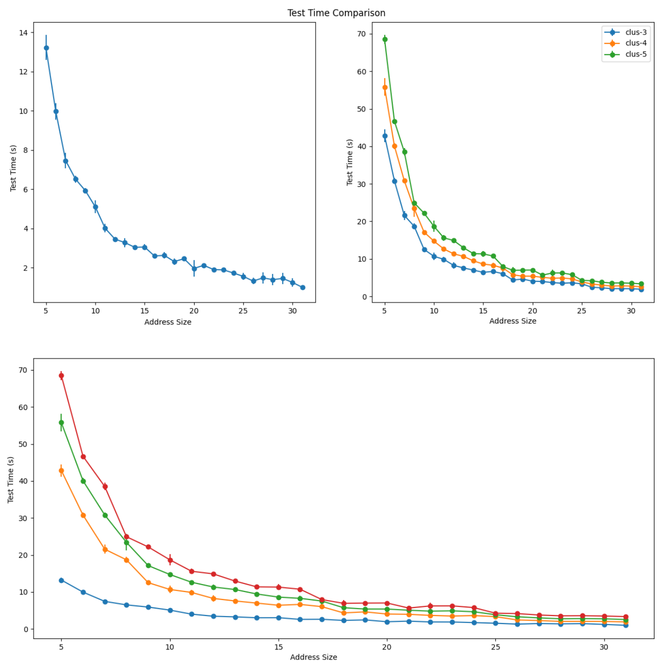


Figure C.30: Comparison of test time between WiSARD and ClusWiSARD in Fashion MNIST dataset with mean threshold.

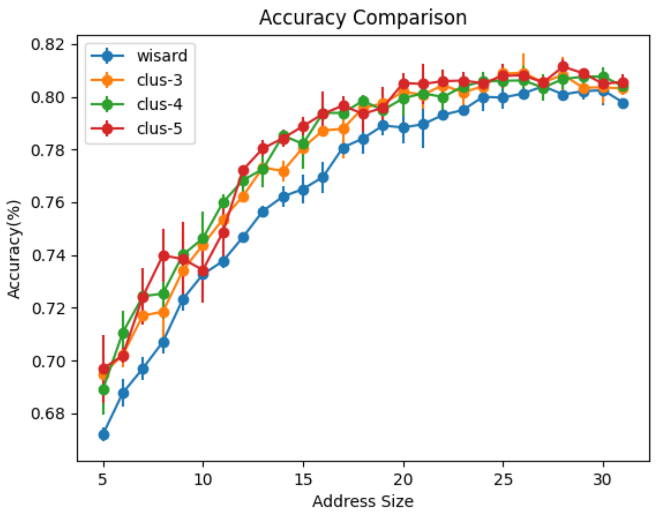


Figure C.31: Comparison of accuracy between WiSARD and ClusWiSARD in Fashion MNIST dataset with Otsu's Binarization.

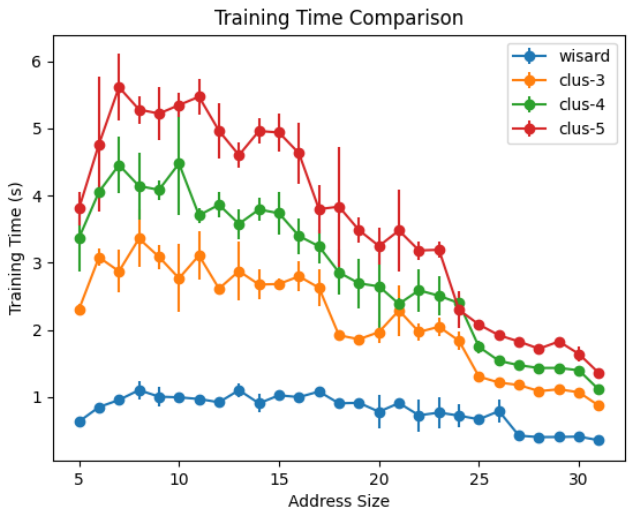


Figure C.32: Comparison of training time between WiSARD and ClusWiSARD in Fashion MNIST with Otsu's Binarization.

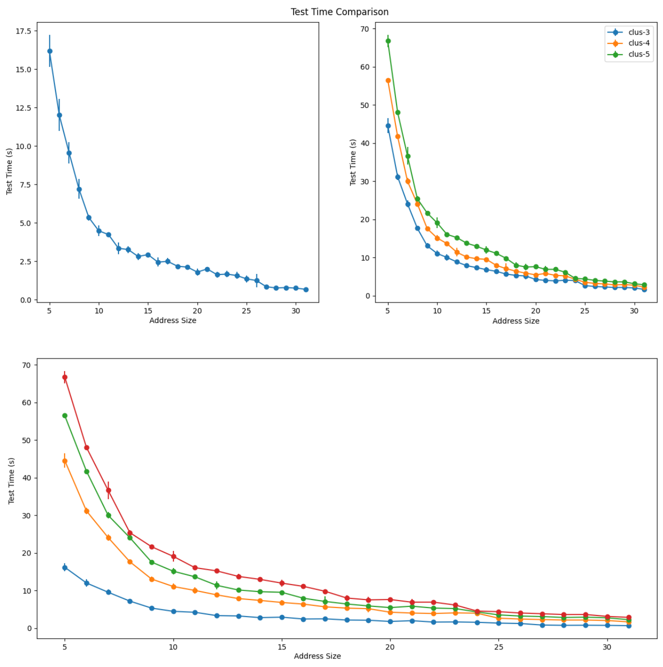


Figure C.33: Comparison of test time between WiSARD and ClusWiSARD in Fashion MNIST dataset with Otsu's Binarization.

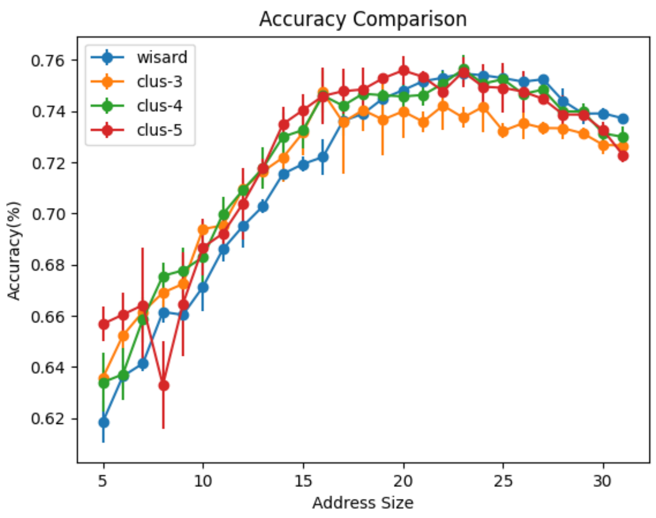


Figure C.34: Comparison of accuracy between WiSARD and ClusWiSARD in Fashion MNIST dataset with Yen's Binarization.

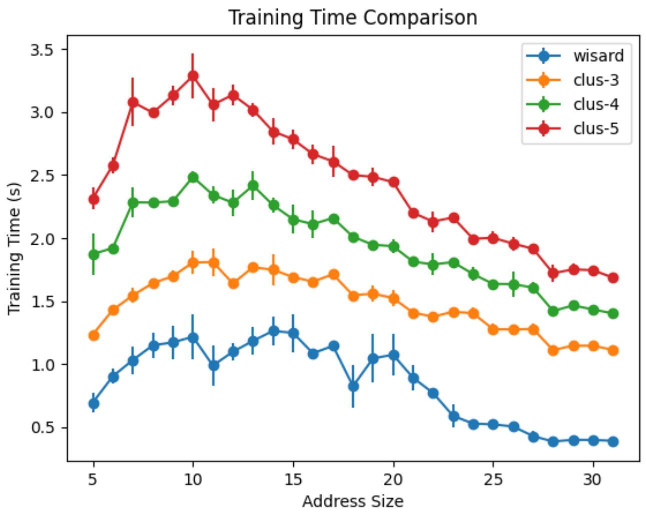


Figure C.35: Comparison of training time between WiSARD and ClusWiSARD in Fashion MNIST with Yen's Binarization.

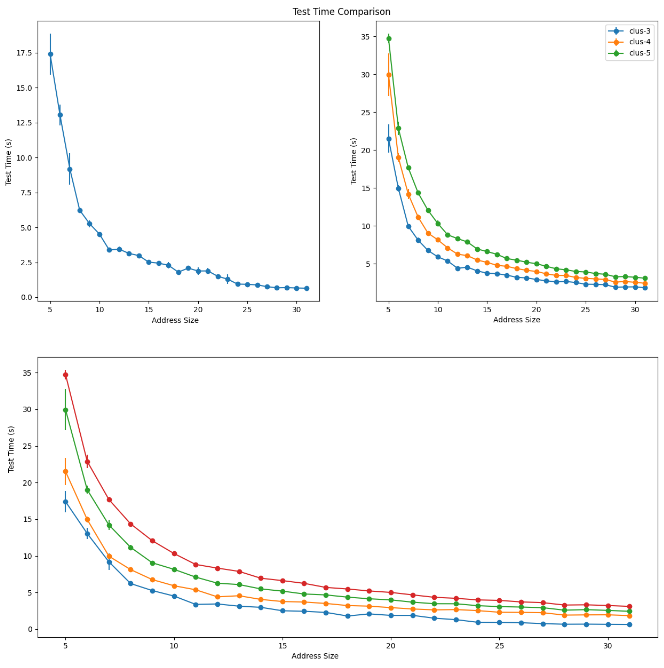


Figure C.36: Comparison of test time between WiSARD and ClusWiSARD in Fashion MNIST dataset with Yen's Binarization.

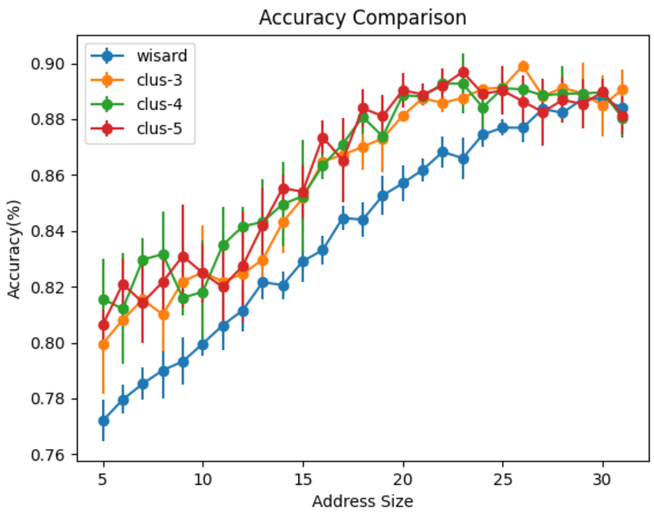


Figure C.37: Comparison of accuracy between WiSARD and ClusWiSARD in MNIST dataset.

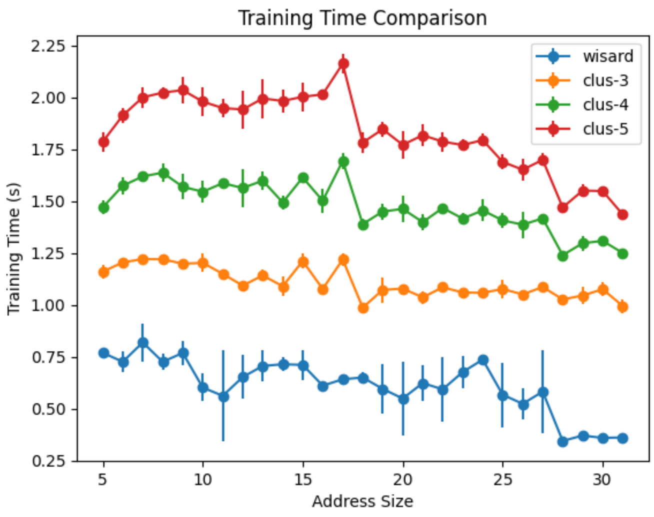


Figure C.38: Comparison of training time between WiSARD and ClusWiSARD in MNIST dataset.

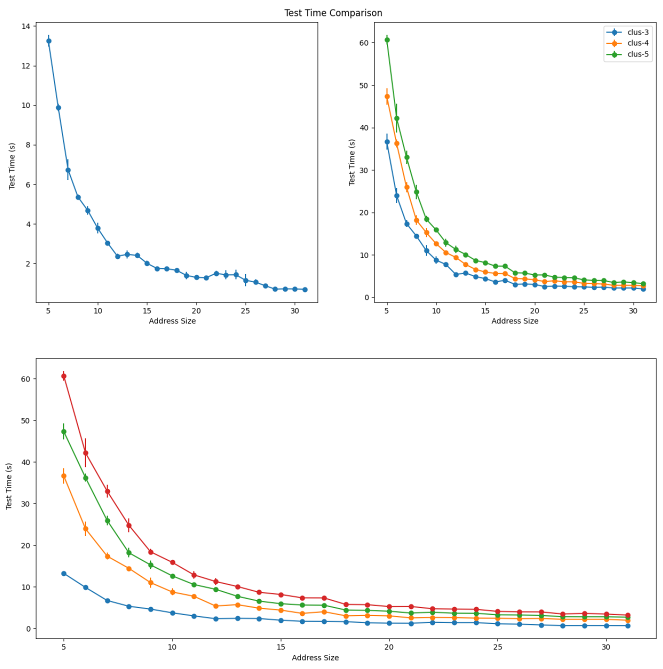


Figure C.39: Comparison of test time between WiSARD and ClusWiSARD in MNIST dataset.

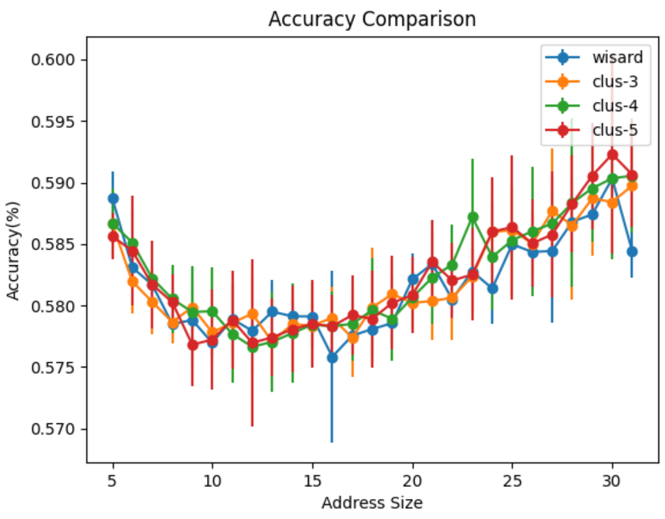


Figure C.40: Comparison of accuracy between WiSARD and ClusWiSARD in IMDB dataset.

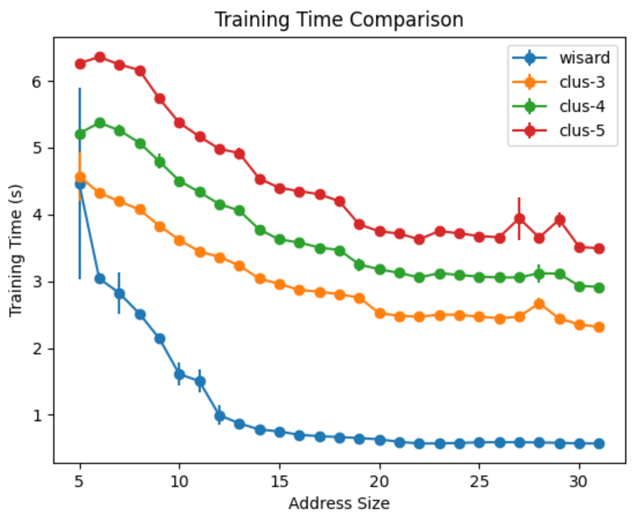


Figure C.41: Comparison of training time between WiSARD and ClusWiSARD in IMDB dataset.

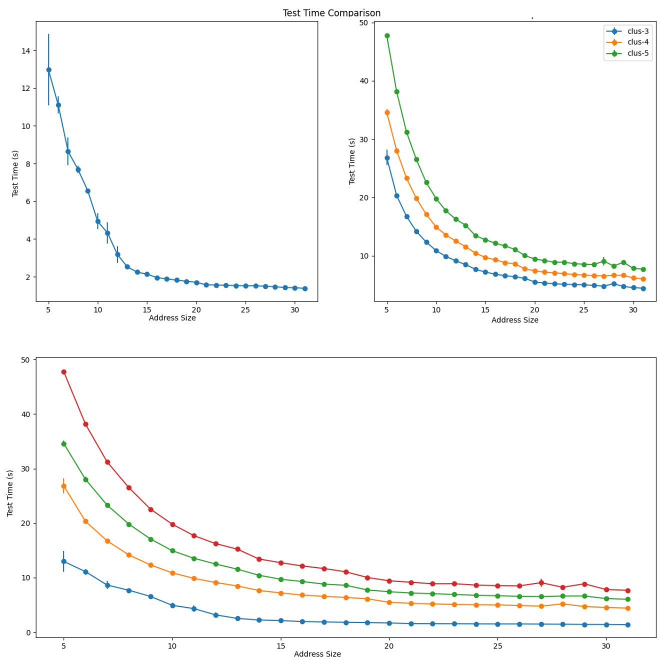


Figure C.42: Comparison of test time between WiSARD and ClusWiSARD in IMDB dataset.

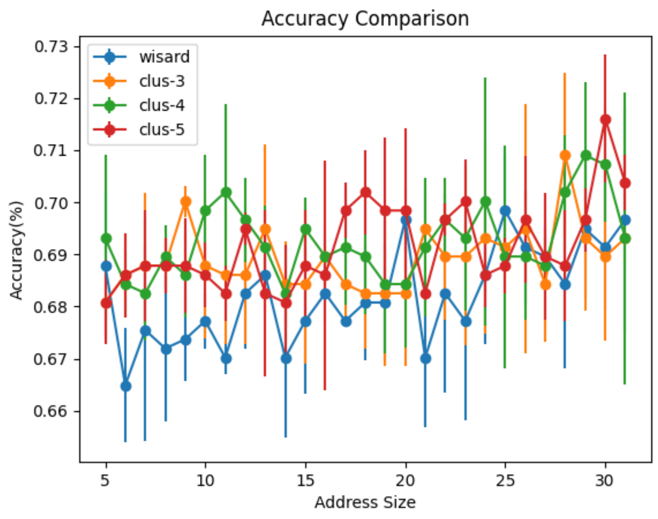


Figure C.43: Comparison of accuracy between WiSARD and ClusWiSARD in MovieLens dataset.

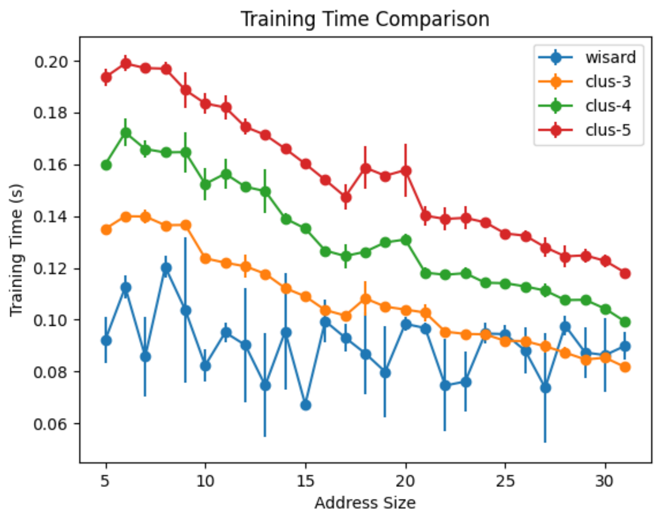


Figure C.44: Comparison of training time between WiSARD and ClusWiSARD in MovieLens dataset.

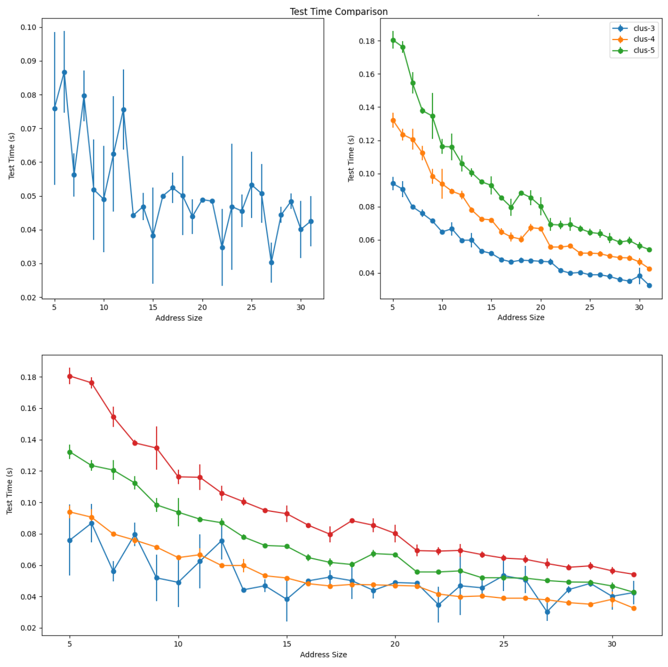


Figure C.45: Comparison of test time between WiSARD and ClusWiSARD in MovieLens dataset.

Appendix D

Supplementary Regression Experiments

In addition to the graphs shown in Chapter 5, others were made to better illustrate the benchmark of the performance of the regression weightless models and those that served as a basis of comparison for them, as they were state-of-the-art in the KDD18 dataset. Here, the metric used was the mean squared error (MSE) to accentuate the difference between the results.

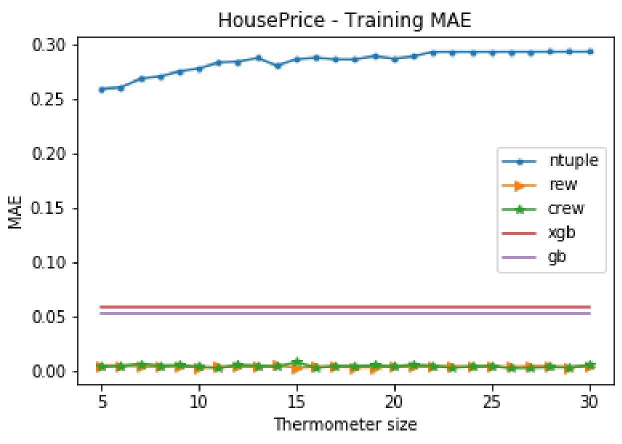


Figure D.1: Thermometer size X MAE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in House Prices.

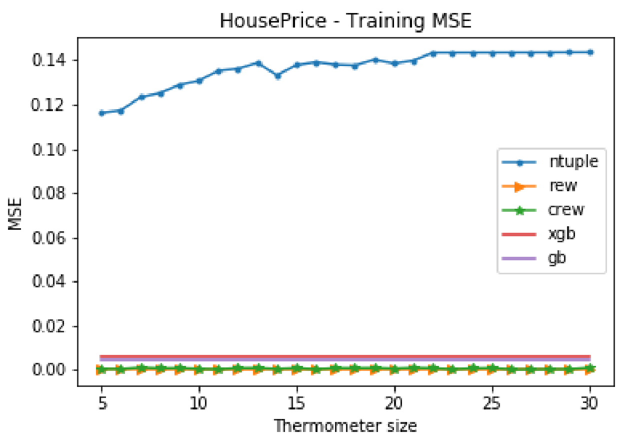


Figure D.2: Thermometer size X MSE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in House Prices.

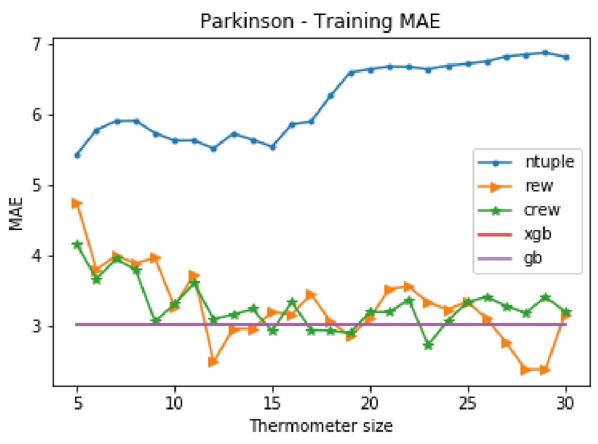


Figure D.3: Thermometer size X MAE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in Parkinson.

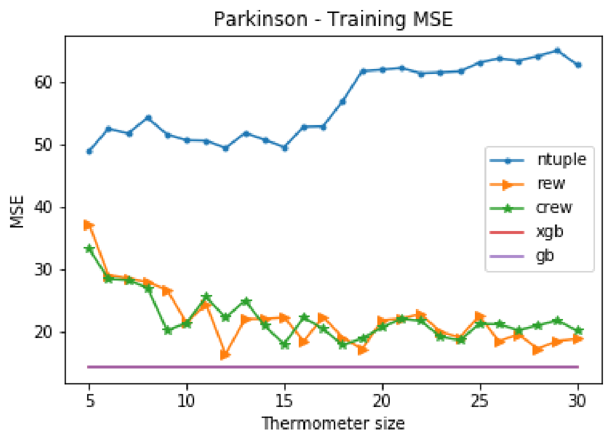


Figure D.4: Thermometer size X MSE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in Parkinson.

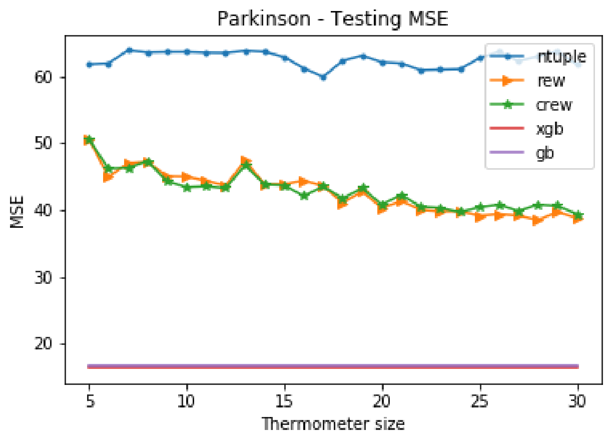


Figure D.5: Thermometer size X MSE (test set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in Parkinson.

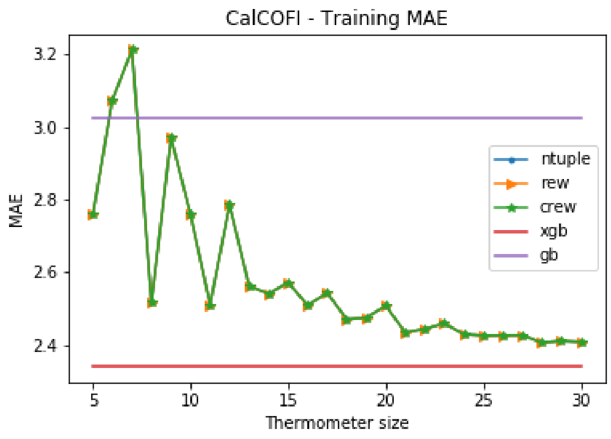


Figure D.6: Thermometer size X MAE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

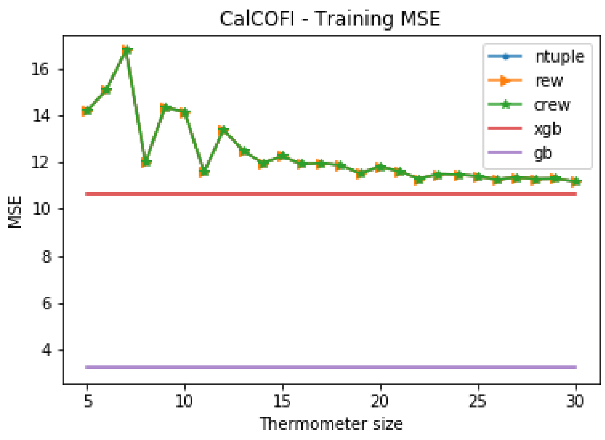


Figure D.7: Thermometer size X MSE (training set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

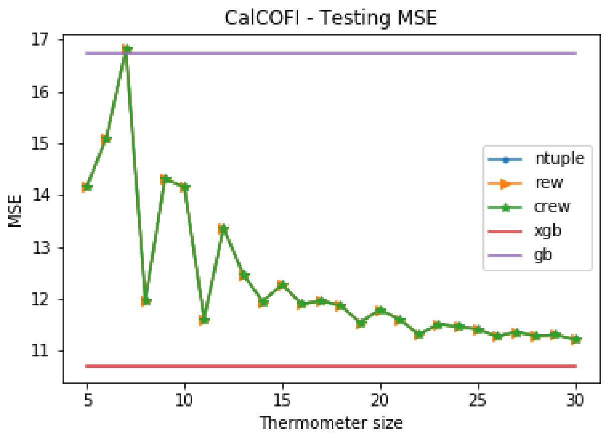


Figure D.8: Thermometer size X MSE (test set) for ReW, CReW, n-Tuple Regression Network, GradientBoost and XGBoost in CalCOFI.

Appendix E

WiSEMAN: A Weightless Emotion-driven Neural Architecture for Planning-related Tasks

From the WiSARD's extensions achieved in this thesis, it was possible to envision a WiSARD-based cognitive architecture. Their motivations and structure are detailed below.

E.1 Motivations

The increasing number of devices collecting data and connecting, exchanging information, makes concepts like IoT and BigData tangible. The emergence of efficient ways to deal with such a huge heterogeneous mass of data in real-time has become extremely necessary. In this scenario, distributed solutions, such as multi-agent planning, have been more and more highlighted. The ability of agents to be adaptive, agile in their negotiations and, able to handle multiple types of tasks simultaneously is of fundamental importance in this type of system. Because of this, different types of cognitive architecture have been designed to structure the learning and action-taking capabilities of such agents[262].

Another area of computing that has gradually developed is Machine Consciousness (MC)[263], which attempts to build systems that have subjectivity. One of MC's partial goals is to construct agents who behave indistinguishably from that of a truly conscious agent, even if unintentionally. In all of these systems, emotion modeling plays a vital role as a tool for improving agent training. This is to so expected because all major theories of consciousness place great emphasis on the role of emotions.

Substantial, though gradual, advances have been made in this type of system, and some prototypes have already been embodied in robots, with motion and visual control systems, sound, touch, and pain receptors[264]. A naval dispatching system[265] was also designed. Based on correlates to consciousness principles it incorporated natural language processing, database interaction, and resource management.

A common limitation of different architectures for conscious agents is a large number of partially independent modules for performing complex tasks involving multi-modal learning. Therefore, we propose a multi-agent task planning system whose architecture is inspired by a cognitive system with conscious-like emotion-driven behavior using only weightless artificial neural networks (WANN): WiSEMAN (WiSARD Emotional Multi-Agent Network). WANNs are especially computationally inexpensive. While the WiSEMAN itself is still theoretical and not yet tested, its different modules have been developed separately and tested in different domains.

E.2 WiSEMAN Architecture

The cognitive architecture of the WiSEMAN is based on Aleksander's Kernel Architecture[12], which aims to create an agent with behavior similar to that of a conscious being, incorporating in all modules the consciousness requirements of the Fundamental Postulate of Consciousness[8]. The schema of this architecture is displayed in E.1. Each WiSEMAN agent has four operational modules:

- **Depiction:** Analogous to the process of empirical recognition of the environment, this module is responsible for interpreting the environment where the agent is. It captures as much information as possible from all sources, and applies some treatment to them, allowing any noise to be minimized. Recent explorations on a video empathy prediction task[5] have shown how effective WiSARD can be in cross-modal learning. This module should contain a collection of binarization operations to be used for the specific domain task. To achieve that, a WiSARD classifier can be used to choose which binarization will be applied to each piece of data. The binarization domains that are already consolidated in WiSARD are image, audio, PoS tagging, tracker, quantitative, and qualitative variables, time-series, unsupervised and clustering tasks.

- **Imagination:** This module proposes possible actions to be taken by the agent, considering its current state and the environment. The module's operation is based on WiSARD's generative potential. Such a capability can easily generate mental images[109] of the learned subject simply by reverse-engineering the training mechanism and mapping the contents of memory locations into new input. Since WiSARD's input is binary, different thresholds can be applied to this new input, thus generating a collection of possible future state possibilities.

The internal operation of this module can be better understood through the equations C.1, C.2, and C.3, where S' is obtained through transformations of mental images of a trained discriminator with known situations.

$$A = agents; s = states; a = actions \tag{E.1}$$

$$s_{A_i} + a_{A_i}xx \mapsto S' \tag{E.2}$$

$$S' \mapsto s'_{a_0} + ... + s'_{a_n} \tag{E.3}$$

- **Emotion:** A quantitative evaluation of the action possibilities generated by the Imagination Module is made here, using a Regression WiSARD-based multidimensional RAM. Thus, for each action performed, a feedback value is assigned to it depending on its outcome for the agent. Different emotions could be modeled here, but for simplicity, the emotional state here is represented by a continuous value on an emotional positive-negative scale. Some polarized emotions that can be used here include joy-distress, hope-fear, and satisfaction-frustration.

- **Attentioon:** In this module, each feature that forms the representation of the actions that can be performed from the states contemplated by the Imagination Module is evaluated separately. In WiSARD model, each of a discriminator's RAMs represents all the possible combinations of values of a certain subset of features. The bits that integrate the tuple that will address a specific memory piece could be chosen *a priori* according to the available knowledge of the domain. This module ultimately penalizes features that do not contribute to efficient decision-making, causing the system to diminish or nullify its contribution to the planning process. The attention reinforcement feedback is displayed in Equation C.4.

$$y = \frac{\sum_{i=0}^{n} E_i * a_i}{\sum_{i=0}^{n} c_i} \tag{E.4}$$

WiSEMAN is designed to solve typical multi-agent planning problems, such as listing process priority in a system, processing data from multiple sensors stored in a BigData repository, decentralized and distributed computing for IoT devices, coordinating a team of robots serving disaster areas, etc. The differential of this architecture is the use of the generative capacity of the network and the emotional reward signals as a way to improve the generation of action proposals to be performed

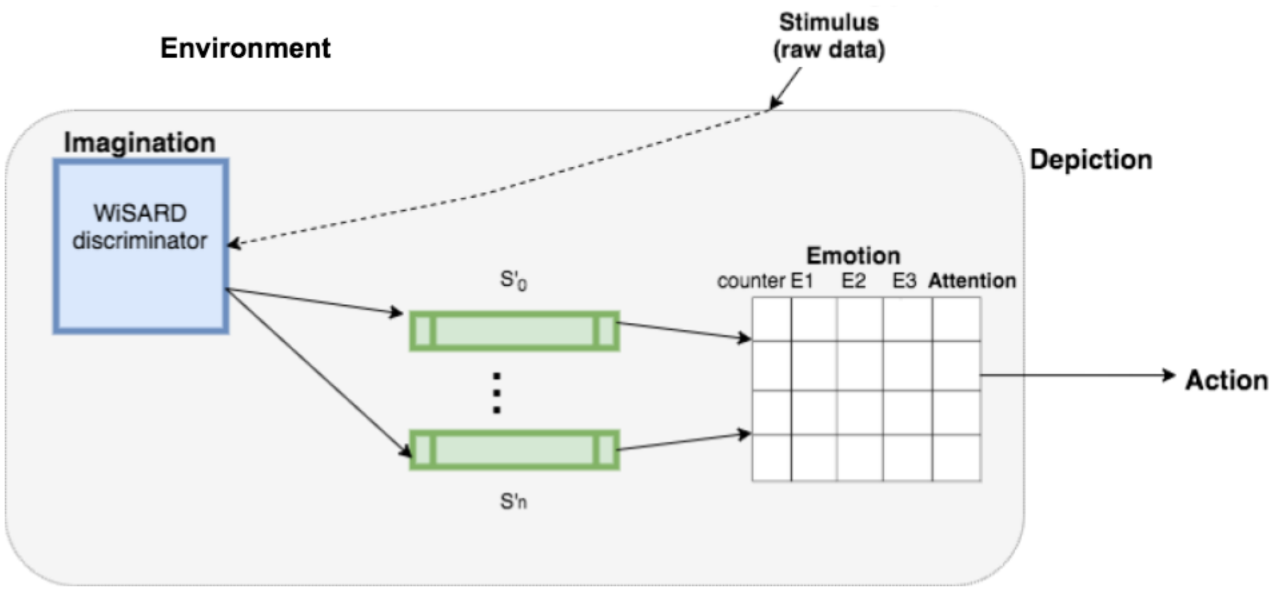


Figure E.1: Architecture of WiSEMAN.

and their consequent evaluation.

Ongoing work includes adding in WiSARD attention mechanisms, multi-dimensional architecture for reinforcement learning, and mapping self-modification system.

Appendix F

Weightless Artificial Neural Networks and Artificial Consciousness

WANN models have been consistently used in experiments related to Artificial Consciousness: brain-inspired architecture for cognitive robotics[16], demonstration of quasi-phenomenal behavior of intelligent agents[13], iconic learning [14, 15], brain structures modeler MAGNUS[167] and validation of global workspace theory using information integration[9–11].

The concept of a theory for artificial consciousness-based only on neural machines was first presented by Professor Igor Aleksander in a paper in 1994[8]. Soon after he formulates the first theory of artificial consciousness based on the personal construct theory[30] of the psychologist George Kelly, formulated in the 1950s, and part of the premise that anticipation and prediction capabilities are the main drivers of mind. Aleksander then formulates his Fundamental Postulate of Consciousness. In the next section, we will replicate the explanation in the entirety of the Fundamental Postulate, as Aleksander originally presented it in [8].

F.1 The Fundamental Postulate: Consciousness and Neural Activity

The personal sensations that lead to the consciousness of an organism are due to the firing patterns of some neurons, such neurons being part of a larger number which form the state variables of a neural state machine, the firing patterns having been learned through a transfer of activity between sensory input neurons and the state neurons.

The words of this postulate are intended to have specific meanings which need to be stressed so that the corollaries which follow should make sense.

- **Personal sensation:** Much of the controversy surrounding consciousness comes from the problem of infinite regress. Here it is implied that neural activity leads directly to personal sensation so dismissing the problem of infinite regress;
- **Firing patterns:** Neurological terminology has been adopted to refer to the output activity of a group of neural elements. In an artificial system 'firing patterns' could refer to any measurement of the output quantity of the elements which constitute that system;
- **Neurons:** This adoption of this neurological term is used to indicate that the theory is that of a cellular system where "neuron" is the name given to a basic cell;
- **Neural state machine:** A state machine is the most general model of a finite computing process - it calls on the concept of an inner state which is a function of input sequences. Neural versions assume that neurons generate the variable values which, when taken together, form a state. (Corollary 1 formalizes this notion and the generality of neural state machines has been argued elsewhere);
- **Learned:** Neurons are assumed to be plastic and it is this plasticity that allows them to learn meaningful, representational, firing patterns;
- **Iconic Transfer:** This key property relates to the source of information that controls the learning of the neurons. It will be seen that distal, sensory information is postulated to impose output patterns on neurons so that these may be learned and recalled in the absence of input. It is this transfer that creates an inner perception in the conscious organism;
- **Sensory Neurons:** These are transducer neurons that transform energy from environmental input into the distal, sensory signals which control iconic transfer.

F.2 Advances in WiSARD and the Investigation in Artificial Consciousness

Aleksander also derives twelve corollaries from the Fundamental Postulate:

- Brain as a state machine;
- Inner Neuron Partitioning;

- Conscious and unconscious states;
- Perceptual learning and memory;
- Prediction;
- Awareness of self;
- Representation of meaning;
- Learning utterances;
- Learning language;
- Will;
- Instinct;
- Emotion.

Considering the use of WiSARD in an architecture that tries to satisfy the Fundamental Postulate and its corollaries, Regression WiSARD directly implies the possibility of satisfying the predictive capacity of the model. The multi-dimensional RAM neuron can also have learning fields and use a feedback function to meet the Will, Instinct, and Emotion corollaries, although such a solution is neurosymbolic and not strictly connectionist, as intended by Aleksander. In addition, ReW's multi-dimensional RAM can be used in a model analogous to MAGNUS[167].

Appendix G

WiSARD Libraries

All experiments in this thesis used the same implementation of WiSARD. In addition to the library containing the models, a library was also used for multi-label classification systems and ensembles.

G.1 wisardpkg

To facilitate the production of codes using WiSARD-based models, LabZero developed an ML library C++/Python called wisardpkg. This library is an MIT-licensed open-source package hosted on GitHub under the license.

G.1.1 Implementation

wisardpkg is hosted on GitHub at https://iazero.github.io/wisardpkg/, where users can find the latest version of the library and user documentation. Model details and features described in this publication pertain to the latest version of the model as of the date of this publication. The lib was implemented in C/C++ with a Python wrapper.

G.1.2 Availability

- **Operating systems:** Linux, Mac OSX, Windows
- **Programming languages:** C++ 11 and up, Python version 3.7.0 and up
- **Additional system requirements:** NA
- **Dependency:** pybind11 ($\geq$ 2.5.0)
- **List of contributors:** All contributors were listed as authors with corresponding affiliations

- **Language:** C++, with wrapper to Python 3
- **Current version:** 2.0.0a7

G.1.3 Installation

- **C++:**
  - Clone https://github.com/IAZero/wisardpkg
  - Copy the wisardpkg.hpp file to the desired project
  - Include the library in the C++ code
- **Python:**
  - Install Python PIP, if necessary
  - pip install wisardpkg

To install wisardpkg in a Windows environment it is necessary to install Visual C++ additionally. To do this just download it from here and then run the installer.

G.1.4 Architecture

The library is divided into two main modules: models and binarization because since these neural networks only receive binary inputs, it is necessary to treat the input to make it suitable for models. Although this is usually done through some kind of preprocessing external to wisardpkg, the library has some classes to provide support for this pipeline.

G.1.4.1 Binarization

All binarization classes are extensions of the BinBase class. All of them receive an array as input to their unique public method, transform, which will return a binary array. Only the public methods of each class will be described here.

Thresholding: applies a simple threshold to a double value to generate a binary input.

- Thresholding: its only parameter is the threshold.
- transform

MeanThresholding: similar to the previous one, but this time the threshold is calculated as the mean of the input data.

- MeanThresholding
- transform

Thermometer: is a technique for preprocessing quantitative variables. Given a variable d, a maximum value of traing test m and a number of ranges s, the new binary variable will have s bits, with each ith bit being determined by a threshold $t = i * \frac{m}{s}$. If $d > t$, the ith position is worth 1, otherwise 0.

- SimpleThermometer: its parameters are the thermometer size, the minimum, and the maximum value in its range.
- transform

KernelCanvas: since each WiSARD-based model can handle only one input size, this preprocessing[130] is capable of resizing inputs, being especially useful when dealing with time series. This uses different kernels, or divisions in the sample space of the input, replacing each value of it with the central value of the kernel where it is located.

- KernelCanvas: it is possible to instantiate it from a JSON file. Its parameters are the desired dimensionality and the number of kernels to be used.
- transform

G.1.4.2 Models

This module contains all the models and also the base classes from which they extend. A brief description of each sub-module and its classes follows.

1. **Base:**
    - **Model:** a simple trainable module
        - train
        - getsizeof
    - **ClassificationModel:** a Model object that can calculate the similarity score in access, as well as perform classifications
        - classify
        - rank
        - score

- **RegressionModel:** a Model object that performs predictions. Here the training is overwritten because of the partial prediction used in learning in this type of model.
  - train
  - predict

2. **Wisard:**

- **RAM:** the minimal information unit in a weightless neural network; contains 2^n memory positions.
  - RAM: instantiates RAM. It is possible to use a JSON file with RAM previously saved for this. Two additional parameters here are ignoreZero (which allows not considering the initial position of each RAM in the classification phase) and base (its default value is 2, forming the classic WiSARD for binary patterns, but when modifying it is possible to work with patterns that use more bits and the RAM will have $base^n$ memory locations).
  - getVote
  - train
  - untrain: it is possible to reverse the training process of an example, once the positions accessed are known, just subtracting their access counters.
  - getMentalImage: using the retina and the content of the RAM, it generates a representation of the learning.
  - setMapping: it is possible to choose a mapping for the RAM.
- **Discriminator:**
  - Discriminator: its main parameter is the size of the tuple. It is also possible to define its mapping and instantiate it from a JSON file. Its main methods are: train, untrain and classify.
- **Wisard:** a ClassificationModel that has a set of discriminators. Its main methods are: train, untrain and classify. An optional parameter "balanced" when set to True causes the score of each discriminator during the classification to be normalized using the number of trained examples.

3. **Cluswisard:** has only one homonymous class, which will be described below:

- Cluswisard: its main parameters are the size of the tuple and the variables used in the verification to create new discriminators: minScore, threshold, and discriminatorsLimit.

- The main methods here include train, untrain, classify, trainUnsupervised, and classifyUnsupervised, the latter is applied only when it is desired to know which is the discriminator with which the example is most similar, despite classes.

4. **RegressionWisard:**

    - **MeanFunctions:** this module has a Mean class, which serves as the basis for several other classes that contain the methods of the means used in the prediction of Regression WiSARD (SimpleMean, PowerMean, Median, HarmonicMean, HarmonicPowerMean, GeometricMean, ExponentialMean, and LogisticMean).
    - **RegressionRAM:** analogous to the classification RAM, it has an extra content in its memory positions, which is the partial prediction. Additional parameters here include minZero and minOne, which are the minimum amount of these bits that a memory location needs to have to be considered in the prediction phase.
    - **RegressionWisard:** the network itself, a set of RegressionRAMs. Its main parameters are the size of the tuple and the average to be used, with minZero, minOne, completeAdressSize, and mapping being additional parameters. Like other Models, it can be instantiated from a JSON file. Its main methods are train and predict.

5. **ClusRegressionWisard:** This module has only one homonymous class, which is a RegressionModel, whose main instantiation parameters are addressSize, minScore, threshold, and limit. Its main methods are train and predict.

Additionally, the library has a commons module, with exceptions and utils, and a wrapper module for Python.

G.2 Multi-label Classification Systems

A library that uses wisardpkg was made in Python for multi-label classification tasks. Here is a description of its classes:

- class CrossValidation:
    - init (parameteres: images, labels, k)
    - validation (parameters: method, metrics)
- class Method():

  - run (parameters: X-train, X-test, y-train, classes)

- class LabelPowerset(Method):
  - run (parameters: X-train, X-test, y-train, classes)

- class BinaryRelevance(Method):
  - run (parameters: X-train, X-test, y-train, classes)

- class ClusLabelPowerset(Method)
  - init (parameters: addr-size, minScore, threshold, discriminatorLimit)
  - run (parameters:X-train, X-test, y-train, classes):

- class ClusBinaryRelevance(Method)
  - init (parameters: addr-size, minScore, threshold, discriminatorLimit)
  - run (parameters: X-train, X-test, y-train, classes)

The classes in this script allow you to use each of the multi-label architectures with WiSARD or ClusWiSARD, in addition to performing k-fold cross-validation with both architectures. The script is hosted on GitHub at https://github.com/thesis-wisard/thesislibs.git.

G.3 Classification Ensembles

All classes in this library have the *classify* method that has as a parameter a wisardpkg.DataSet object.

The Bagging class is instantiated as the method *init(train-dataset, learners, partitions, models)*, where *train-dataset* is a wisardpkg.Dataset object, *learners* is a number of weak learners in the ensemble, *partitions* is a percentage of examples in training set that will be used in composition of each training subset already considering the resampling, *models* can assume the values ["wisard", "cluswisard", "heterogeneous"].

The Boost class is instantiated as the method *init(train-dataset, validation-dataset, learners, models = "heterogeneous")*, where *train-dataset* and *validation-dataset* are wisardpkg.Dataset objects, *learners* is a number of weak learners in the ensemble, *models* can assume the values ["wisard", "cluswisard", "heterogeneous"].

The Borda class is instantiated as the method *init(train-dataset, learners, partitions, voting)*, where *train-dataset* is a wisardpkg.Dataset object, *learners* is a number of weak learners in the ensemble, *partitions* is a percentage of examples in training set that will be used in composition of each training subset already considering the resampling, *voting* is a classification policy and can assume the values ["borda0", "borda1", "dowdall"].

The VotingBagging class encapsules Tie-break and Weighted Votes ensembles. This is instantiated as the method *init(train-dataset, learners, partitions, voting)*, where *train-dataset* is a wisardpkg.Dataset object, *learners* is a number of weak learners in the ensemble, *partitions* is a percentage of examples in training set that will be used in composition of each training subset already considering the resampling, *voting* is a classification policy and can assume the values ["plurality1", "plurality2", "plurality3", "plurality4"], "plurality1" employs Tie-break ensemble with "All candidates" policy, "plurality2" employs Tie-break ensemble with "Only tiers", "plurality3" employs Weighted Votes ensemble and "plurality4" employs Tie-break ensemble with "Threshold" policy.

This code is hosted on GitHub at https://github.com/thesis-wisard/thesislibs.git.

Appendix H

List of Publications

List of publications produced during the making of this thesis. All of them are directly or indirectly linked to the theme of this thesis.

H.1 Journal Articles

1. L. A. D. Lusquino Filho, L. F. R. Oliveira, A. S. Lima Filho, G. P. Guarisa, L. M. Felix, P. M. V. Lima and F. M. G. França, *Extending the Weightless WiSARD Classifier for Regression*, Neurocomputing, Special Issue: ESANN 2019, 2020.

H.2 Book Chapters

1. L. A. D. Lusquino Filho, L. F. R. Oliveira, H. C. C. Carneiro, G. P. Guarisa, A. S. Lima Filho, P. M. V. Lima and F. M. G. França, *A Weightless Neural System for Empathy Prediction*, OMG Challenge Book, Springer, Edited by Pablo Barros and Prof. Stefan Wermter, **Submitted**.

H.3 Complete Works Published in Proceedings of Conferences

1. L. A. D. Lusquino Filho, L. F. R. Oliveira; A. S. Lima Filho, G. P. Guarisa, P. M. V. Lima and F. M. G. França, *Prediction of palm oil production with an enhanced n-Tuple Regression Network*, Proceedings of the 27nd European Symposium on Artificial Neural Networks, Bruges, Belgium, pp. 301–306, 2019.

2. L. A. D. Lusquino Filho, G. P. Guarisa, L. F. R. Oliveira, A. S. Lima Filho, F.

M. G. França and P. M. V. Lima, *Action Units Classification Using ClusWiSARD*, Proceedings of the International Conference on Artificial Neural Networks, 2019, ICANN 2019: Image Processing, Munich, Germany, pp. 409–420, 2019.

3. A. S. Lima Filho, G. P. Guarisa, L. A. D. Lusquino Filho, L. F. R. Oliveira, C. Cosenza, F. M. G. França and P. M. V. Lima, *Interpretation of Model Agnostic Classifiers via Local Mental Images*, Proceedings of the 28nd European Symposium on Artificial Neural Networks, Bruges, Belgium, 2020.

4. Luiz C. S. Ramos, L. A. D. Lusquino Filho, F. M. G. França and P. M. V. Lima, *Static and dynamic malware detection using weightless neural networks*, 20th Brazilian Symposium on Information Security and Computational Systems, Rio de Janeiro, Brazil, 2020.

5. M. A. Spalenza, E. S. Oliveira, L. A. D. Lusquino Filho, F. M. G. França and P. M. V. Lima, *Using NER + ML to Automatically Detect Fake News*, Proceedings of the International Conference on Intelligent Systems Design and Applications, online, 2020.

H.4 Extended Abstracts Published in Proceedings of Conferences

1. L. A. D. Lusquino Filho, A. S. Lima Filho, F. M. G. França and P. M. V. Lima, *WiSEMAN: A weightless emotion-driven neural architecture for planning-related tasks*, Workshop CogArch, Proceedings of the 26th IEEE International Symposium on High-Performance Computer Architecture, San Diego, USA, 2020.

2. L. A. D. Lusquino Filho, L. F. R. Oliveira, H. C. C. Carneiro, G. P. Guarisa, A. S. Lima Filho, F. M. G. França and P. M. V. Lima, *A weightless regression system for predicting multi-modal empathy*, Workshop Affective Behavior Analysis in-the-wild, Proceedings of the 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), Buenos Aires, Argentina, 2020.

3. L. A. D. Lusquino Filho, F. M. G. França and P. M. V. Lima, *Exploring WiSARD-based Models in the Building of Cognitive Systems for Emotive Perception*, Workshop on Consciousness, Models and the Artificial, Proceedings of Creativity 2019, Rio de Janeiro, Brazil, pp. 141, 2019.

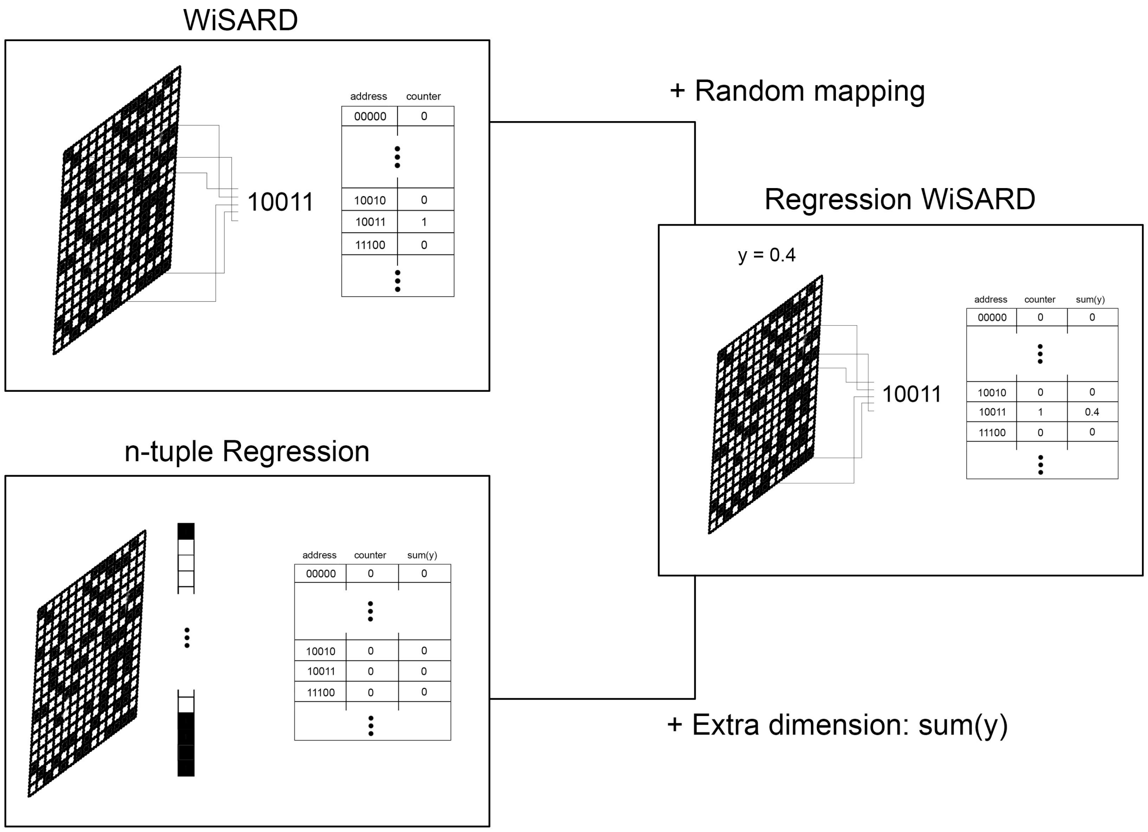
4. R. N. C. B. Rocha, L. A. D. Lusquino Filho, M. Aredes, F. M. G. França and P. M. V. Lima, *Regression WiSARD application of controller on DC STATCOM converter under fault conditions*, 9th Workshop on Parallel Programming Models, Proceedings of the 34th IEEE Parallel and Distributed Processing (IDPDS 2020), New Orleans, USA, 2020.

5. A. S. Lima Filho, L. A. D. Lusquino Filho, F. M. G. França and P. M. V. Lima, *"What are you Thinking?": Explanation and Interpretation by an Artificial Consciousness System*, Workshop on Consciousness, Models and the Artificial, Proceedings of Creativity 2019, Rio de Janeiro, Brazil, pp. 139–140, 2019.

Graphical Abstract

Extending the Weightless WiSARD Classifier for Regression

Leopoldo A. D. Lusquino Filho[1*], Luiz F. R. Oliveira[1*], Aluizio Lima Filho[1], Gabriel P. Guarisa[1], Lucca M. Felix[2], Priscila M. V. Lima[1,3], Felipe M. G. França[1]
1 - PESC/COPPE; 2 - DCC; 3 - NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil



*Both authors had equal participation and are first author.

Extending the Weightless WiSARD Classifier for Regression

Leopoldo A. D. Lusquino Filho[1*], Luiz F. R. Oliveira[1*], Aluizio Lima Filho[1], Gabriel P. Guarisa[1], Lucca M. Felix[2], Priscila M. V. Lima[1,3], Felipe M. G. França[1]
1 - PESC/COPPE; 2 - DCC; 3 - NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract

This paper explores two new weightless neural network models, Regression WiSARD and ClusRegression WiSARD, in the challenging task of predicting the total palm oil production of a set of 28 (twenty eight) differently located sites under different climate and soil profiles. Both models were derived from Kolcz and Allinson's n-Tuple Regression weightless neural model and obtained mean absolute error (MAE) rates of 0.09097 and 0.09173, respectively. Such results are very competitive with the state-of-the-art (0.07983), whilst being four orders of magnitude faster during the training phase. Additionally the models have been tested on three classic regression datasets, also presenting competitive performance with respect to other models often used in this type of task.

Keywords: Regression WiSARD WiSARD ClusWiSARD n-Tuple classifier n-Tuple Regression Network Ensemble Online learning

1. Introduction

Regression is a traditional and important machine learning task, since there is a wide range of practical situations in the real world where it is necessary to predict values in a continuous space. In a precision agriculture scenario, it would be desirable that simple devices, such as small sensors,

[*]Both authors had equal participation and are first author.

A Weightless Neural System for Empathy Prediction

Leopoldo A. D. Lusquino Filho[⋆], Luiz F. R. Oliveira, Hugo C. C. Carneiro, Gabriel P. Guarisa, Aluzio Lima Filho, Priscila M. V. Lima, and Felipe M. G. França

PESC/COPPE/Universidade Federal do Rio de Janeiro, RJ, Brazil
leopoldolusquino@gmail.com, lfdeoliveira@cos.ufrj.br, hcesar@cos.ufrj.br, gabrielguarisa@gmail.com, lima.filho.a.s@gmail.com, priscilamvl@gmail.com, felipe@cos.ufrj.br

Abstract. Affective computing encompasses far more than just the evaluation of emotions of individuals. One challenge faced in this field is the estimation of attributes derived from interactions among different agents, such as the empathy experienced in interpersonal relationships. This work takes into account the benefits of machine learning in order to estimate the valence of emotions on the OMG Empathy dataset, considering the information obtained from face expressions and dialogue of interlocutors. RegressionWiSARD and ClusRegressionWiSARD *n*-tuple regressors were employed to this end. A vital part of the experiments reported here involved a discussion on efficient ways to preprocess heterogeneous data and how to convert them into relevant binary input to RAM-based networks. The best performance achieved among all the combinations of weightless neural models considered (evaluated using the CCC metric) was 0.25 and 0.015 on the Personalized Track in validation and test set, respectively.

Keywords: weightless neural network; n-tuple regression; WiSARD; affective computing; empathy prediction

1 Introduction

The ability to experience and evaluate the degree of empathy is a vital skill for human survival and the construction of its social interactions. Empathy prediction is, consequently, an area of great interest within Affective Computing, since many other computational tools can take great profit from it, for instance virtual tutors, security systems, and dynamic customization of applications oriented to the detection of instant emotions.

A usual way of categorizing emotions is through valence and arousal attributes, both assuming continuous values. Valence refers to how pleasant a particular feeling was (a negative valence refers to an unpleasant event) and arousal to how intense it was. A slightly happy situation would have a positive and high

[⋆] Corresponding author

Prediction of Palm Oil Production with an Enhanced n-Tuple Regression Network

Leopoldo A. D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Aluizio L. Filho[1], Gabriel P. Guarisa[1], Priscila M. V. Lima[2], Felipe M. G. França[1] *

1- PESC/COPPE 2- NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract. This paper introduces Regression WiSARD and ClusRegression WiSARD, two new weightless neural network models that were applied in the challenging task of predicting the total palm oil production of a set of 28 differently located sites under different climate and soil profiles. Both models were derived from the n-tuple regression weightless neural model and obtained error (MAE) rates of 0.08737% and 0.08938%, respectively, which are very competitive with the state-of-art (0.07569), whilst being four (4) orders of magnitude faster during the training phase.

1 Introduction

Regression is one of the most important machine learning tasks, given the wide range of practical situations in the real world where it is necessary to predict values in a given continuum space. Due to its great utility, it is desirable that simple devices, such as small sensors, could perform regression with online training. Weightless artificial neural networks (WANNs), due to its lean, RAM-based architecture, seems to be ideal for this type of task.

This paper presents and explores the use of WANNs in the KDD18 competition [5], a challenge which goal is to predict the palm oil harvest productivity of a set of 28 different production fields using data provided by an agribusiness company. The dataset contains information about palm trees varieties, harvest dates, atmospheric data during the development of the trees, and soil characteristics of the fields where the trees are located in. The novel WANN models are based on the n-tuple Regression Network [3], which has been proved successful when compared to other classical regression approaches in non-linear plant approximation, and Mackey-Glass chaotic time series prediction tasks.

The remainder of this text is organized as follows: Section 2 presents the two weightless models proposed for regression, as well as the basic concepts behind the models that inspired it: WiSARD [1] and n-tuple Regression Network. Section 3 discusses the various approaches used in the KDD18 competition, as well as a comparison with state-of-the-art methods and other relevant results. Conclusion and future work are presented in Section 4.

*This work was partially supported by CAPES, CNPq, FAPERJ and FINEP, Brazilian research agencies.

Action Units Classification using ClusWiSARD

Leopoldo A. D. Lusquino Filho[0000-0002-8283-3764][1*], Gabriel P. Guarisa[1], Luiz F. R. Oliveira[1], Aluizio Lima Filho[1], Felipe M. G. França[1], and Priscila M. V. Lima[12]

1- PESC/COPPE 2- NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil [**]
lusquino@cos.ufrj.br,gabrielguarisa@gmail.com,aluizio@cos.ufrj.br,
lfdeoliveira@cos.ufrj.br,felipe@cos.ufrj.br,priscilamvl@gmail.com

Abstract. This paper presents the use of WiSARD and ClusWiSARD weightless neural networks models for the classification of the contraction and extension of *Action Units*, the facial muscles involved in emotive expressions. This is a complex problem due to the large number of very similar classes, and because it is a multi-label classification task, where the positive expression of one class can modify the response of the others. WiSARD and ClusWiSARD solutions are proposed and validated using the CK+ dataset, producing responses with accuracy of 89.66%. Some of the major works in the field are cited here, but a proper comparison is not possible due to a lack of appropriate information about such solutions, such as the subset of classes used and the time of training/testing. The contribution of this paper is in the pioneering use of weightless neural networks in an AUs classification task, in the unpublished application of the WiSARD and ClusWiSARD models in multi-label tasks and in the new unsupervised expansion of ClusWiSARD proposed here.

Keywords: Action Units, WiSARD, ClusWISARD, weightless neural network

1 Introduction

Ekman and Friesen [10] cataloged a set of muscles known as *Action Units* (AUs) – which would be responsible for all facial expressiveness – while attempting to obtain a set of universal emotions present in any human. The automatic identification of these AUs has been developed since the mid-1990s and has several applications: forensics, psychological treatment, physical therapy support and advertising feedback, among others. AUs have also been used in the development of adaptive digital avatars [4].

Some of the great difficulties in automatic detection of AUs are the large number of classes and the wide variety of forms how AUs express themselves,

[*] Corresponding author

[**] This work was partially supported by CAPES, CNPq, FAPERJ and FINEP, Brazilian research agencies.

Interpretation of Model Agnostic Classifiers via Local Mental Images

Aluizio Lima Filho[1], Gabriel P. Guarisa[1], Leopoldo A.D. Lusquino Filho[1],
Luiz F. R. Oliveira[1], Carlos A. N. Cosenza[3],
Felipe M. G. França[1] and Priscila M. V. Lima[1,2] *

1–PESC/COPPE, 2–NCE, 3–PEP/COPPE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract. Although successful black-box learning models have been created, understanding what happens when a machine produces a classification response is still a challenge. This work introduces FRWI – Fuzzy Regression WiSARD Interpreter, a novel fuzzy rules-based algorithm that is capable of interpreting the responses of black-box classifiers via the production of local mental images from a WiSARD n-tuple classifier. FRWI is compared with LIME – Local Interpretable Model-Agnostic Explanations, a pioneering agnostic classification interpreter model. To make a quantitative evaluation of interpretable models, a new metric – Interpretation Capacity Score – is proposed. Using this metric, it is shown that FRWI surpasses LIME in producing coherent interpretations.

1 Introduction

The need to interpret responses from learning models gets higher in different situations [1]. Questions arise such as: how the models make the decision in the classification, or when to trust its process, and when not to do so. One way to answer the first question is to show what is relevant to the model. LIME [2] – Local Interpretable Model-Agnostic Explanations – was developed with the motivation to clarify such relevance. There are other interpreter models focused on DNNs, like Gran-Cam [3], that were later introduced in the literature. How ever, LIME does not have feasible interpretation capacity for all learning models, due to interpretable models have scenarios where they work better as learning models. Experimental tests were performed utilizing LIME to explain decisions made by following classifiers: WiSARD [4], Linear model [5] and Random Forest model [6] trained with images data sets. It will be shown that results will select too much in the image as relevant, and it will not let it clear what is happening inside the classifier.For that reason, the idea of creating a degree of relevance for each pixel in the image came as an alternative to interpret the responses of black-box classifiers more feasible. This work introduces FRWI – Fuzzy Regression WiSARD Interpreter, a WiSARD n-tuple classifier that produces local mental images, via a fuzzy rules-based algorithm, as an interpretation of the responses of black-box classifiers. To compare the interpretation capacity of both LIME and FRWI models, the Interpretation Capacity Score metric is defined.

*This work was partially supported by NGD Systems, Inc./COPPETEC grant PESC21713; CAPES, CNPq and FAPERJ, Brazilian research agencies.

Detecção estática e dinâmica de *malwares* usando redes neurais sem peso

Luiz C. S. Ramos[1], **Leopoldo A. D. Lusquino Filho**[1], **Felipe M. G. França**[1], **Priscila M. V. Lima**[1]

[1]Programa de Engenharia de Sistemas e Computação (PESC/COPPE)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{sampaio,lusquino,felipe,priscilamvl}@cos.ufrj.br

Resumo. *A preocupação com a segurança e a integridade dos dados em sistemas de computação, incluindo áreas importantes como Internet das Coisas e Indústria 4.0, estão crescendo dramaticamente. Dessa forma, a existência de um malware pode ameaçar o bom funcionamento de sistemas inteiros, trazendo consequências irreversíveis. Este trabalho visa utilizar redes neurais sem peso para detecção estática e dinâmica de malwares. Na utilização de técnicas estáticas baseadas na imagem 2D do arquivo binário e de técnicas dinâmicas baseadas em API Calls, a rede WiSARD mostrou resultados próximos de técnicas do estado da arte utilizando redes neurais com peso, porém com tempos de treinamento e classificação uma ordem de grandeza menor.*

Abstract. *Concerns with security and integrity of data in computer systems, including important areas such as Internet of Things and Industry 4.0, are dramatically increasing. Therefore, the existence of malware can threaten the smooth functioning of whole systems, bringing irreversible consequences. This work aims to use weightless neural networks for static and dynamic detection of malwares. Using static techniques based on the 2D image of binary files and dynamic techniques based on API Calls, the WiSARD network showed results close to state-of-the-art techniques using convolutional neural networks, but shorter training and classification times one lower order of magnitude. The method presented shows the WiSARD as a faster alternative in detecting malware.*

1. Introdução

Malwares, ou *softwares* maliciosos, são programas que buscam perturbar as operações de um sistema computadorizado por meio de acesso não autorizado para conseguir informações sensíveis, ameaçando usuários [Aycock 2006]. Esses programas podem comprometer a integridade das informações nesse sistema, de forma que a detecção de *malwares*, ou seja, técnicas para identificar se um programa é malicioso ou não, é alvo de pesquisadores, já que novos *malwares* constantemente são lançados e as técnicas devem acompanhar tais mudanças [Bazrafshan et al. 2013].

Com a criação constante de novos programas maliciosos, além dos métodos tradicionais de detecção de *malwares*, como *signature-based methods* e *behavior-based methods*, métodos baseados em heurística, utilizando técnicas de aprendizado de máquinas, como Naïve Bayes [Schultz et al. 2001] começaram a ser explorados

Using NER + ML to Automatically Detect Fake News

Marcos A. Spalenza[1], Elias de Oliveira[1], Leopoldo Lusquino-Filho[3], Pricila M. V. Lima[2], Felipe M. G. França[3]

[1] Postgraduate Program in Informatics – Federal University of Espírito Santo (UFES)
[2] NCE, Tércio Pacitti Institute – Federal University of Rio de Janeiro (UFRJ)
[3] PESC/COPPE – Federal University of Rio de Janeiro (UFRJ)
marcos.spalenza@gmail.com, elias@lcad.inf.ufes.br

Abstract.

In the overload information era, we need to be conscious of the dissemination of incoherent and misleading content both in the traditional and social media. It is a problem that has worsened recently and called the attention of some governments worldwide. The so-called Fake News has got notoriety due to the popularization and rapid consumption of online news. The democratization of the internet access carried out an increase in independent production and consumption of a variety of unverified information contents, which are also spread around on a large scale. Because the production capacity is much higher than that of the fact-checking agencies, it becomes necessary the support of systems for automatic detection of this type of content. Therefore, in this article, we propose a linguistic-structure analysis approach with named-entity recognition to identify fake news. By applying our approach, we can identify linguistic-structures that must unveil an article produced and verified by professional news agencies from that false information and sensationalist. In this regard, we present a linguistic analysis system with 90% on average accuracy of identification surpassing the state-of-the-art of this type of content in the literature datasets.

Keywords: Fake News, Named-Entity Recognition, W*i*SARD, Machine Learning, Artificial Intelligence

1 Introduction

Content's revision and verification processes have long been used for the control, organization, and search of documents organization even before their publication. This documentary analysis consists of textual reviews, format and stylistic validation, cohesion, and contextualization guarantees. Among these methods, the automatically textual review has been continuously improving, and thus, computer systems became an important tool to aid in pointing out inconsistencies in the textual sequence. Thereby the area of Natural Language Processing (NLP) turns itself fundamental for content checking and ensuring consistency.

Among other challenges in the analysis of written language, the detection of false news has recently taken on great social appeal. Such an appeal occurs due to the social-political impact in several countries worldwide and, consequently, gains legislative notoriety within these countries [25]. In Brazil, a recent Fake News Inquiry by Federal Supreme Court (STF) demonstrates the relevance of this debate, both at popular and

WiSEMAN: A weightless emotion-driven neural architecture for planning-related tasks

Leopoldo A.D. Lusquino Filho, Aluizio Lima Filho,
Felipe Maia Galvão França, Priscila Machado Vieira Lima

PESC/COPPE
Universidade Federal do Rio de Janeiro – Brazil

ABSTRACT

Planning and management systems via multi-agent have become an increasingly demanding solution for many tasks involving heterogeneous data. Research into conscious agents has also shown solid progress. Here we propose an agent-based cognitive system with conscious-like behavior using Weightless Artificial Neural Networks. WiSEMAN is easy to embed on a wide range of devices due to low computational cost.

1. INTRODUCTION

The increasing number of devices collecting data and connecting to each other, exchanging information, makes concepts like IoT and BigData tangible. The emergence of efficient ways to deal with such huge heterogeneous mass of data in real time has become extremely necessary.In this scenario, distributed solutions, such as multi-agent planning, have been more and more highlighted. The ability of agents to be adaptive, agile in their negotiations and able to handle multiple types of tasks simultaneously is of fundamental importance in this type of system. Because of this, different types of cognitive architecture have been designed to structure the learning and action taking capabilities of such agents[1].

Another area of computing that has gradually developed is Machine Consciousness (MC)[2], which attempts to build systems that have subjectivity. One of MC's partial goals is to construct agents who behave indistinguishably from that of a truly conscious agent, even if unintentionally. In all of these systems, emotion modeling plays a vital role as a tool for improving agent training. This is to so expected, because all major theories of consciousness place great emphasis on the role of emotions.

Substantial, though gradual, advances have been made in this type of system, and some prototypes have already been embodied in robots, with motion and visual control systems, sound, touch and pain receptors[3]. A naval dispatching system[4] was also designed. Based on correlates to consciousness principles it incorporated natural language processing, database interaction, and resource management.

A common limitation of different architectures for conscious agents is the large number of partially independent modules for performing complex tasks involving multi-modal learning. Therefore, we propose a multi-agent task planning system whose architecture is inspired by a cognitive system with conscious-like emotion-driven behavior using only weightless artificial neural networks (WANN): WiSEMAN (WiSARD Emotional Multi-Agent Network). WANNs are especially computationally inexpensive. While the WiSEMAN itself is still theoretical and not yet tested, its different modules have been developed separately and tested in different domains.

Section 02 describes the WiSARD, WANN model used here, and Section 03 details the WiSEMAN architecture and refers to the basis on the technology needed to build its modules.

2. WISARD

WANNs are RAM-based neural networks where each neuron is a simple memory table. Learning on these models consists on memory writes, and classification on memory reads. A traditional and simple model of WANN is WiSARD[5], a class discriminator-oriented architecture that has an input retina, responsible for performing a pseudo-random mapping of n-tuples of a binary input into specific RAM locations. Since N neurons compose each WiSARD discriminator, the length of a binary input is roughly $N * n$ bits. Each memory location stores an integer that represents the number of hits in the memory position addressed by the n bits in question.

The WiSARD training phase consists of feeding an input data to the model and increment the counter inside each accessed RAM location of a class discriminator. To classify a sample, all discriminators access their RAM locations and return the counters that represent the scores of each class. The discriminator with the highest score is the predicted class of the model. If a score tie occurs, a threshold technique called *bleaching* is applied. The *bleaching* value is initialized to zero and

A weightless regression system for predicting multi-modal empathy

Leopoldo A. D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Hugo C. C. Carneiro[1], Gabriel P. Guarisa[1], Aluízio Lima Filho[1], Felipe M. G. França[1] and Priscila M. V. Lima[1 2]
[1] PESC/COPPE, [2] NCE - Universidade Federal do Rio de Janeiro, RJ, Brazil

***Abstract*— This work takes into account the benefits of machine learning in order to estimate the valence of emotions on the OMG Empathy dataset, considering the information obtained from face expressions and dialogue of interlocutors. RegressionWiSARD and ClusRegressionWiSARD n-tuple regressors and its ensembles were employed to this end. The best performance achieved among all the combinations of weightless neural models considered (evaluated using the CCC metric) was 0.25 in validation set of the Personalized Track .**

I. Introduction

Since emotional states are a fundamental part of the core of human psychology, often exceeding the intellect itself in psychological hierarchy, it is natural that Affective Computing[11] occupies a prominent place in the study of the human-machine interface. Along with the apogee of machine learning, Affective Computing has experienced great growth in recent years, but it still has many open questions. Some of the main ones involve the prediction of emotions based on information from many different sources and the identification of subtle emotional states in real time. Specifically, many advances have been made recently in the area of affective prediction[25][26][27][28][29][30][31][32].

In order to offer a significant contribution in the area, this paper discusses the use of weightless neural network ensembles in predicting the affective valence of individuals in conversation videos, since this type of model has computational simplicity, great computational agility and ease of being parallelized.

The structure of this work is as follows: in Section 2 the WiSARD weightless model, some of its extensions and ensembles will be described, Section 3 deals with the preprocessing of data from different sources, Section 4 describes the experiments carried out with different types of ensembles using uni and multimodal data and discusses their results, and Section 5 is the conclusion of this work, summarizing all the previous discussion and also offering the main ongoing works.

II. n-Tuple models

The n-Tuple classifier is a boolean node pattern classifier [3], which distances itself from models derived from perceptron because it do not use synaptic weights between their neurons, thus avoiding all training time required for their convergence. n-Tuple classifier does not need any parameter fine tuning, nor does it use any error minimization technique to obtain generalization in pattern learning [17]. The family of models derived from the n-Tuple classifier is known as Weightless Artificial Neural Network (WANN).

This work was not supported by any organization

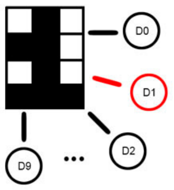


Fig. 1. The WiSARD model multidiscriminator structure. For digits recognition task there are ten discriminators. In the training phase, only the corresponding discriminator is accessed.

A. *WiSARD*

WiSARD is a neural model based on the n-tuple classifier, where each neuron is equivalent to a piece of memory [1]. This model is class discriminator-oriented, where all discriminators are formed by N RAM-neurons, whose memory addresses are addressed by n-bit tuples. Each neuron has 2^n memory locations.

WiSARD works with binary standards, requiring the use of some preprocessing technique to form data suitable to the model before the training and classification process. The training process consists of using the binary input to access specific memory positions of the corresponding discriminator and increment the counter that constitutes its content. During the classification, all discriminators are accessed and they are assigned a score formed by the number of non-null positions accessed. The discriminator with the highest score will determine the class of the entry and in case of a tie, a threshold called bleaching, which is initialized to zero, is increased and the classification is repeated, considering for the score only memory locations whose counter has higher value than bleaching. This procedure is repeated until there is a winning discriminator or until the bleaching value exceeds the highest counter among the memory locations accessed, in which case a default class is chosen for the entry. The structure and the training process in WiSARD are illustrated in Figs. 1 and 2. WiSARD can be used to accelerate the training of deep models, and can be used as a starting layer for such neural networks in a hybrid hierarchy[33].

Exploring WiSARD-based models in the building of cognitive systems for emotive perception

Leopoldo A.D. Lusquino Filho, Priscila Machado Vieira Lima, Felipe Maia Galvão França

PESC/COPPE
Universidade Federal do Rio de Janeiro – Brazil

Abstract. Recently, the weightless neural network model WiSARD has been used in different tasks in the emotional domain, such as the classification of facial emotions (Lusquino Filho et al., 2018), Action Units detection and multi-modal prediction of empathy in dialogues (Lusquino Filho et al., 2019), thus demonstrating its ability to recognize continuous and discrete emotional states. Since emotion plays a vital role in the cognitive process (Megill, 2003; Denton et al., 2009; Damasio and Carvalho, 2013), this was a significant contribution to extend WiSARD to make it suitable for Artificial Consciousness (AC) frameworks.

Weightless artificial neural networks (WANN) are neural models that do not use weighted synapses to store the information it learns from presented patterns. On the other hand, it possesses RAM(random-access-memory)-based neurons in which this storage takes place. In a WANN the learning of a pattern simply corresponds to writing in memory, whereas classification essentially corresponds to the reading of certain memory positions. WiSARD is the pioneering model of WANNs and since its creator, Igor Aleksander, is also one of the pioneers of the study of AC, WiSARD and other WANN models have been consistently used in experiments related to AC: brain-inspired architecture for cognitive robotics (Shanahan, 2005), demonstration of quasi-phenomenal behavior of intelligent agents (Aleksander and Morton, 2009) and validation of global workspace theory using information integration (Aleksander and Gamez, 2009; Aleksander and Gamez, 2011).

Nevertheless, limitations in WiSARD's architecture prevent it from satisfying the 12 corollaries of the Fundamental Postulate of Artificial Neuroconsciousness, derived from Kelly's "artificial constructs" theory (brain as a state machine; conscious and unconscious states; perceptual learning and memory; prediction; awareness of self; representation of meaning; learning utterances; learning language; will; instinct; and emotion).

However, new extensions to WiSARD, such as DRASiW (Grieco et al., 2010), ClusWiSARD (Cardoso et al., 2016) and Regression WiSARD (Lusquino Filho et al., 2019) have added capabilities in the model to represent its own learning, cluster data and perform linear and logistic regression, allowing it to more fully satisfy the 12 corollaries. These are the works we intend to present here.

Regression WiSARD application of controller on DC STATCOM converter under fault conditions

Raphael N. C. B. Rocha [†], Leopoldo L. Filho[‡], Mauricio Aredes[†], Felipe M. G. França[‡], Priscila M. V. Lima[‡§]
[†]*Programa de Engenharia Elétrica – COPPE*
[‡]*Programa de Engenharia de Sistemas e Computação – COPPE*
[§]*Tercio Pacitti Institute – CCMN*
Universidade Federal do Rio de Janeiro – UFRJ
Rio de Janeiro, Brazil
raphael.dkreng@poli.ufrj.br, lusquino@cos.ufrj.br, aredes@lemt.ufrj.br, felipe@cos.ufrj.br, priscilamvl@cos.ufrj.br

***Abstract*—Capable of supplying local loads, DC microgrids have received much attention in the last decade for alleviating power flow through the main power grid. This has been achieved through the use of edge devices on the control of the converters, but, among other problems, microgrids have stability issues when Constant Power Loads (CPL) are present. This problem was already solved in the literature with the DC STATCOM power converter, in normal operation mode, it can deal with the grid operation. However, in fault cases, the solutions available still fail to ignore faults or even contribute to them. The present work aims to explore the potential of a light machine learning algorithm of the type Weightless Artificial Neural Network (WANN) for predicting the output of the original controller used in the DC STATCOM on an edge device connected to a converter, and investigate its generalization capability under microgrid fault situations. The WANN used is based on the regression variant of the Wilkes, Stonham, and Aleksander Recognition Device (WiSARD), coined as Regression WiSARD (ReW). The evaluation criteria employed measured the capability of the controller to reject the fault condition. Initial results showed surprisingly good results in comparison to the original DC STATCOM controller, indicating that a ReW-based controller plays well the role of the DC STATCOM and was able to cope with fault situations.**

***Index Terms*—weightless artificial neural networks, Regression WiSARD, smart grids, DC microgrids, edge computing.**

I. Introduction

Distributed Generation (DG) is a topic of increasing interest to the global energy sector. DC microgrid [1] constitutes a way to interconnect several DG's, as well as Energy Storage Systems (ESS), where dc loads can be also connected to. Each power converter heavily depend on edge devices since all data processing related to the converters must be performed locally. To further explore the fault handling capabilities of a machine learning-based converter, one must also comply with these requirements.

An example of dc microgrid is shown in Fig. 1, which comprise some DG, ESS, and local loads. Usually, a dc microgrids is connected to an ac power grid through a dc/ac converter know as Grid-Connected Converters (GCC) that can act as a grid former (if it provides voltage and frequency control), or as grid follower, depending on the power rating of that converter with respect to the installed power generation in the ac grid.

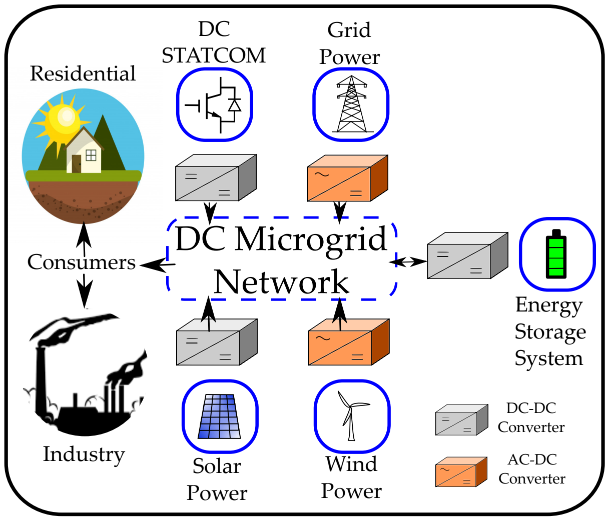


Fig. 1: DC microgrid connected to a conventional grid and to micro generators

A critical problem associated with dc microgrid control is the difficulty of controlling the dc bus voltage in the presence of dc loads that behave as Constant Power Loads (CPL) [2] [3], because the non linearity of these loads may render the system unstable. There are different ways of treating this problem, one of them is to employ a DC compensator associated with an energy storage system connected at the Point of Common Coupling (PCC). The use of such DC/DC compensator can bring benefits tho the DC microgrid, in analogy to the STATCOM and active filters that are capable of enhancing the quality of power, or improving the stability margin in AC grid/microgrids [4]. It is intuitive to assume that the same concept can be applied to DC microgrids. The Direct Current Static Compensator (DC STATCOM) was proposed by Chen [5] aiming at compensating theses instabilities on the DC microgrids, as depicted in Fig. 2.

However, DC STATCOMs can cause problems in case of fault in one of the loads, generators or transmission lines

“What are you thinking?” - Explanation and Interpretation by a consciousness system

Aluizio Lima Filho, Leopoldo A.D. Lusquino Filho
Priscila Machado Vieira Lima, Felipe Maia Galvão França
PESC/COPPE
Universidade Federal do Rio de Janeiro – Brazil

In the XAI(eXplainable Artificial Intelligence) area the focus is on the explanation of what is happening inside the AI(Gunning, 2017). For make that possible is needed generate some interpretation of the AI behaviour(Ribeiro et al., 2016).
The explanation is a process that involve the argumentation and emotion response. The target is to persuade the person who receive the arguments(Molnar, 2019).
In the that process there are two ways of interpretations: the first one is the interpretation made by the explainer to create the arguments; the second one is the interpretation made by who receive the argumentation to comprehend what has been said.
In a consciousness system, it can have your consciousness of its existence and consciousness of its perceptions and so on(Alexander, 1995), and the explanation helps in the consciousness of itself.
The ability to create explanations is needed to make the system aware of itself and helps the system to create logical “thoughts”.
The consciousness of itself could be made possible when the system generates interpretations of what it percepts and with that create its argumentation to start the process of explanation, and therefore it creates a line of reasoning that make it to start to “think” or just to ratiocinate.
The first steps is to create tools capable of generate interpretation as resource to the consciousness system. In the on going research of interpretations of the model WiSARD(Aleksander et al., 1984) a tool capable of generate images as interpretation resources could be used to the consciousness system.

Bibliography

Gunning, David. "Explainable artificial intelligence (xai)." *Defense Advanced Research Projects Agency (DARPA), nd Web* 2 (2017).

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier." *Proceedings of*