UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

RODOLFO ANÍBAL LOBO CARRASCO

Hypercomplex-Valued Recurrent Neural Networks and their Applications for Image Reconstruction and Pattern Classification

Redes Neurais Recorrentes Hipercomplexas e suas Aplicações em Reconstrução de Imagens e Classificação de Padrões

Campinas

2021

Rodolfo Aníbal Lobo Carrasco

Hypercomplex-Valued Recurrent Neural Networks and their Applications for Image Reconstruction and Pattern Classification

Redes Neurais Recorrentes Hipercomplexas e suas Aplicações em Reconstrução de Imagens e Classificação de Padrões

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Matemática Aplicada.

Thesis presented to the Institute of Mathematics, Statistics and Scientific Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Applied Mathematics.

Supervisor: Marcos Eduardo Ribeiro do Valle Mesquita

Este trabalho corresponde à versão final da Tese defendida pelo aluno Rodolfo Aníbal Lobo Carrasco e orientada pelo Prof. Dr. Marcos Eduardo Ribeiro do Valle Mesquita.

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

L786h

Lobo Carrasco, Rodolfo Aníbal, 1989-
Hypercomplex-valued recurrent neural networks and their applications for image reconstruction and pattern classification / Rodolfo Aníbal Lobo Carrasco. – Campinas, SP : [s.n.], 2021.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Redes neurais recorrentes. 2. Redes neurais de Hopfield. 3. Memória associativa. 4. Redes neurais hipercomplexas. I. Mesquita, Marcos Eduardo Ribeiro do Valle, 1979-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Redes neurais recorrentes hipercomplexas e suas aplicações em reconstrução de imagens e classificação de padrões
Palavras-chave em inglês:
Recurrent neural networks
Hopfield neural networks
Associative memory
Hypercomplex-valued neural networks
Área de concentração: Matemática Aplicada
Titulação: Doutor em Matemática Aplicada
Banca examinadora:
Marcos Eduardo Ribeiro do Valle Mesquita [Orientador]
Guilherme de Alencar Barreto
Peter Sussner
Carlile Campos Lavor
Fernando Maciano de Paula Neto
Data de defesa: 16-04-2021
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-9600-8469
- Currículo Lattes do autor: http://lattes.cnpq.br/8527212680361762

Tese de Doutorado defendida em 16 de abril de 2021 e aprovada

pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARCOS EDUARDO RIBEIRO DO VALLE MESQUITA

Prof(a). Dr(a). GUILHERME DE ALENCAR BARRETO

Prof(a). Dr(a). PETER SUSSNER

Prof(a). Dr(a). CARLILE CAMPOS LAVOR

Prof(a). Dr(a). FERNANDO MACIANO DE PAULA NETO

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Acknowledgements

I want to thank my couple Paula, my family and friends, from all over the world, that I made in these years, for the continuous support, love and company. I dedicate this work especially to those who left us along this path, not only during this process in particular, but rather, throughout the long journey in the world of science: Norma Cubillos, Jaime Alfaro, my Kuky and my Panchita. We didn't have a goodbye but they are in my thoughts and feelings. I also want to deeply thank Professor Marcos Valle, for his patience, support and understanding. I would like to emphasize that I greatly appreciate his human quality and the neatness of his work, which never disappeared and which was incredibly delicate in difficult moments. This work is also the result of his constant company as a counselor, being for me the greatest scientific teacher with whom I have had the privilege of learning and whom I deeply admire. I hope there are many opportunities for our other works "to shine" and other spaces so that we can talk and say goodbye. Besides, I want to thank all the teachers of the Institute of Mathematics and Scientific Computing, with whom I was formed and reformed in the science of mathematics. In particular to Professor João Meyer for his classes, advice and talks, and to Professor Sandra Santos for helping me in my first contacts with the Institute, giving me the opportunity to take this beautiful road. I hope that one day my country will provide free and quality education, so that other young people can have the opportunity that you have given me. As I also expect diversity, inclusion and respect for all people, in particular for those most discriminated and segregated. I can't help writing that to have seen Natives, Afro-descendants and many different cultural expressions in the classrooms and patios of UNICAMP, enriched me and made me believe that such a place is possible. As a consequence of the above, I thank all the study groups and teachers of Mapudüngun, who have been and are a new company and a bridge to new ways of thinking, to create new realities in our wounded country. Finally, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brazil (CAPES) - Finance code 001.

Resumo

As redes neurais hipercomplexas, incluindo redes neurais complexas, quaterniônicas e octoniônicas, podem tratar dados multidimensionais como uma única entidade. Em particular, as versões hipercomplexas do modelo de Hopfield foram extensivamente estudadas nos últimos anos. Esses modelos podem apresentar baixa capacidade de armazenamento e efeito de interferência cruzada quando são implementados usando a regra de Hebb. As redes recorrentes por correlação (RCNNs) e suas generalizações para sistemas hipercomplexos são modelos alternativos, que podem ser usados para implementar memórias associativas de alta capacidade de armazenamento. Nesta tese de doutorado, analisamos a estabilidade das redes recorrentes por correlação, usando uma ampla classe de sistemas numéricos hipercomplexos (HRCNNs). Em seguida, fornecemos as condições necessárias para garantir que uma RCNN hipercomplexa sempre estabiliza-se em um ponto de equilíbrio usando os modos de atualização síncrono ou assíncrono. Exemplos com RCNNs bipolares, complexas, hiperbólicas, quaterniônicas e octoniônicas são dados para ilustrar os resultados teóricos. Apresentamos também as redes neurais de projeção recorrente quaterniônicas (QRPNNs). Resumidamente, as QRPNNs são obtidas combinando o aprendizado por projeção (QHPNNs) com a rede recorrente por correlação quaterniônica (QRCNNs). Mostramos que as QRPNNs superam o problema de interferência cruzada das QRCNNs. Assim, elas são apropriadas para implementar memórias associativas. Para validar os resultados teóricos, implementamos memórias associativas para realizar o armazenamento e recuperação de padrões sintéticos, bem como imagens coloridas. Experimentos computacionais revelam que as QRPNNs exibem maior capacidade de armazenamento e tolerância a ruídos do que os modelos QRCNN correspondentes. Também exploramos computacionalmente a extenção dos modelos RPNNs para serem utilizados em uma ampla classe de números hipercomplexos (HRPNNS). Implementamos experimentos computacionais para analisar e comparar o comportamento da dinâmica destes novos modelos com os modelos HRCNNs. Finalmente, apresentamos um novo meta-algoritmo utilizando RCNNs como instância de votação em um *ensemble* de classificadores para resolver problemas de classificação, fornecendo uma ponte entre as RCNNs e *ensemble* de classificadores, mostrando com exemplos computacionais o potêncial de aplicação. Em particular, o modelo exponencial RCNN superou aos classificadores tradicionais *AdaBoost, gradient boosting e random forest* em problemas de classificação binária.

Palavras-chave: Rede neural recorrente; rede Hopfield; memoria associativa; rede neural hipercomplexa.

Abstract

Hypercomplex-valued neural networks, including complex, quaternion and octonion-valued neural networks, can treat multi-dimensional data as a single entity. In particular, hypercomplex-valued versions of the Hopfield model have been extensively studied in the past years. These models might be affected by low storage capacity and cross-talk effect when they are implemented using the Hebbian rule. Another important model is the recurrent correlation neural networks (RCNNs) and their generalizations to hypercomplex systems. They are alternative models, which can be used to implement high capacity associative memories. In this doctoral thesis, we study two discrete recurrent hypercomplex-valued models. First, we analyze the stability of the recurrent correlation neural network defined on a broad class of hypercomplex number systems. Then, we provide the necessary conditions which ensure that a hypercomplex-valued RCNN (HRCNNs) always settles at an equilibrium using either synchronous or asynchronous update modes. Examples with bipolar, complex, hyperbolic, quaternion and octonion-valued RCNNs are given to illustrate the theoretical results. Besides, we introduce the quaternion-valued recurrent projection neural networks (QRPNNs). Briefly, QRPNNs are obtained by combining the non-local projection learning with the quaternion-valued recurrent correlation neural networks (QRCNNs). We show that QRPNNs overcome the cross-talk problem of QRCNNs. Thus, they are appropriate to implement associative memories. To validate the theoretical results, we implement associative memories for the storage and recall of synthetic patterns as well as real color images. Computational experiments reveal that QRPNNs exhibit greater storage capacity and noise tolerance than their corresponding QRCNNs. Besides, we explore the extension of the RPNN models into a broad class of hypercomplex numbers (HRPNNs), implementing computational experiments to analyze and compare the dynamical behavior of these new models with the HRCNNs. Finally, we present a new meta-algorithm using the RCNNs as a voting step for classification problems, providing a bridge between RCNNs and ensemble classifiers. Computational experiment confirm the potential application of the new ensemble classifiers. In particular, the exponential RCNN-based ensemble outperformed the traditional AdaBoost, gradient boosting, and random forest classifiers for binary classification tasks.

Keywords: Recurrent neural network; Hopfield network; associative memory; hypercomplex-valued neural network.

List of Figures

Figure 1 – Diagram that outlines the organization of the thesis. 22

Figure 2 – Main references and the contribution of our work. 24

Figure 3 – Hopfield neural network diagram. 26

Figure 4 – Geometrical example for the csgn function with $K = 4$. 30

Figure 5 – The network topology of quaternionic recurrent correlation and projection neural networks. 41

Figure 6 – Recall probability of bipolar associative memories by the noise intensity introduced in the input vector. For each level of noise the experiment is repeated 100 times, and is calculated the probability of recalling a stored item. 47

Figure 7 – Recall probability of exponential bipolar RPNN and RKAM by the noise intensity introduced in the input vector. 48

Figure 8 – Histogram of the Lagrange multipliers of a RKAM model. 49

Figure 9 – Recall probability of quaternion-valued associative memories by the noise intensity introduced in the input vector. 50

Figure 10 – Original color image, input image corrupted by Gaussian noise with standard deviation 0.1, and the corresponding images retrieved by the quaternion-valued associative memories. 52

Figure 11 – Recall probability of quaternion-valued associative memories by the standard deviation of the Gaussian noise introduced in the input. 53

Figure 12 – Error between the desired output and the retrieved quaternion-valued vector by the standard deviation of the Gaussian noise introduced in the input. . . . 54

Figure 13 – Dynamic of the complex-valued multistate ECNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes. 63

Figure 14 – Evolution of the energy of the complex-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ with a resolution factor $K = 256$. The curves marked by "●" and "○", "■" and "□", as well as "▾" and "▽" refer to pairs of ECNNs that differ only from the update mode. 64

Figure 15 – Dynamic of the hyperbolic-valued multistate ECNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes. 66

Figure 16 – Evolution of the energy of the hyperbolic-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ with a resolution factor $K = 256$. 67

Figure 17 – Dynamic of the hyperbolic-valued ECNN with $\alpha = \beta = 1$ and the activation function $\overline{\text{csgn}}(p) = \text{csgn}(\bar{p})$ using a) synchronous and b) asynchronous update modes. 68

Figure 18 – Evolution of the energy of the quaternion-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ and $K = 16$. The curves marked by "●" and "○", "■" and "□", as well as "▾" and "▽" refer to pairs of ECNNs that differ only from the update mode. 70

Figure 19 – Evolution of the energy of the octonion-valued ECNN with the split sign function designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$. The curves marked by "●" and "○", "■" and "□", as well as "▾" and "▽" refer to pairs of ECNNs that differ only from the update mode. 73

Figure 20 – a) Original (undistorted) gray-scale image, b) input image corrupted by Gaussian noise with standard variation 100, images retrieved by c) bipolar ECNN, d) complex-valued multistate ECNN with $K = 246$, e) quaternion-valued multistate ECNN with $K = 16$, and split-sign octonion-valued ECNN using asynchronous update. 75

Figure 21 – Probability of successful recall by the standard variation of Gaussian noise introduced into the input image. 76

Figure 22 – Dynamic of the complex-valued multistate EPNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes. 77

Figure 23 – Dynamic of the hyperbolic-valued multistate RPNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes. 77

Figure 24 – Dynamic of the hyperbolic-valued EPNN with $\alpha = \beta = 1$ and the activation function $\overline{\text{csgn}}(p) = \text{csgn}(\bar{p})$ using a) synchronous and b) asynchronous update modes. 78

Figure 25 – Histogram of the numbers of iterations for complex-valued multi-state ECNN and EPNN models in synchronous update mode, using $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$, and a resolution factor $K = 256$. The corrupted input vector is obtained by replacing, with probability $\texttt{pr} \in [0.2; 0.9]$, a component of a fundamental memory $\mathbf{u}^1$ by an element from $\mathcal{S}$ taken at random with uniform probability. 79

Figure 26 – Distribution of the numbers of iterations for the quaternion-valued multistate ECNN and EPNN models in synchronous update mode, using $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$, and resolution factor $K = 16$. The corrupted input vector is obtained by replacing, with probability $\texttt{pr} \in [0.2; 0.9]$, a component of a fundamental memory $\mathbf{u}^1$ by an element from $\mathcal{S}$ taken at random with uniform probability. 80

Figure 27 – a) Original (undistorted) gray-scale image, b) input image corrupted by Gaussian noise with standard variation 100, retrieved images by c) bipolar EPNN, d) complex-valued multistate EPNN with $K = 246$, e) quaternion-valued multistate EPNN with $K = 16$, and split-sign octonion-valued EPNN using asynchronous update. 81
Figure 28 – Probability of successful recall by the standard variation of Gaussian noise introduced into the input image, using ECNN and EPNN models. 82
Figure 29 – Training for the RCNN ensemble method. 86
Figure 30 – Storing the output generated by the base classifiers over the training data $\mathcal{T}$ and the input samples X for each h_ξ with $\xi = 1, \ldots, P$. 87
Figure 31 – Configuration of the fundamental memory matrix U and the input vector. The colors green and cyan match with the corresponding dimensions between the columns of U and the input vector $\mathbf{z}(0)$ and the output $\mathbf{y}$. 87
Figure 32 – Hasse diagram of Wilcoxon signed-rank test with a confidence level at 95%. 91
Figure 33 – Box-plot of the average time for prediction of batch of input samples. 91
Figure 34 – Box-plot of the normalized F-measures produced by the ensemble classifiers. 92
Figure 35 – Normalized box-plot of the kappa index for the multi-class ensemble classifiers. 95
Figure 36 – Hasse diagram of Wilcoxon signed-rank test with a confidence level at 95% for the multi-class ensemble classifiers. 96

List of Tables

Table 1 – Absolute error between the fundamental memory $\mathbf{u}^1$ and either the input or the quaternion-valued vector recalled by an associative memory model. . . . 53
Table 2 – Multiplication tables for complex numbers and quaternions. 56
Table 3 – Parameters of the hypercomplex-valued ECNN models designed for the storage and recall of CIFAR images where $N_b = 8192$ and $N_c = N_q = N_o = 1024$. 75
Table 4 – Mean and standard deviation of the F-measures produced by ensemble classifiers using stratified 10-fold cross-validation. 90
Table 5 – Mean and standard deviation of the Kappa Index produced by multi-class ensemble classifiers using stratified 10-fold cross-validation. 95

List of abbreviations and acronyms

ANN	Artificial Neural Network
AM	Associative Memory
HNN	Hopfield Neural Network
RCNN	Recurrent Correlation Neural Network
RKAM	Recurrent Kernel Associative Memory
RPNN	Recurrent Projection Neural Network
CvRCNN	Complex-Valued Recurrent Correlation Neural Network
CHNN	Complex-Valued Hopfield Neural Network
QRCNN	Quaternion-Valued Recurrent Correlation Neural Network
CV-QHNN	Continuous-Valued Quaternionic Hopfield Neural Network
QRPNN	Quaternion-Valued Recurrent Projection Neural Network
QHNN	Quaternion-Valued Hopfield Neural Network
HRCNN	Hypercomplex-Valued Recurrent Correlation Neural Network
(H)ECNN	(Hypercomplex-Valued) Exponential Correlation Neural Network
(H)EPNN	(Hypercomplex-Valued) Exponential Recurrent Projection Neural Network
CvECNN	Complex-Valued Exponential Recurrent Correlation Neural Network
QvECNN	Quaternion-Valued Exponential Recurrent Correlation Neural Network
Re-AHN	Real Part Associative Hypercomplex Number System
VSLI	Very Large Scale Integration
CIFAR	Canadian Institute For Advanced Research
SVD	Singular Value Decomposition
SVM	Support Vector Machines

List of symbols

W	Synaptic Weight Matrix
a_i	Activation potential of the ith neuron
w_{ij}	Weight of the synaptic connection between the jth and the ith neurons
w_{ij}^p	Weight of the synaptic connection between the jth and the ith neurons, obtained by projection rule
w_{ij}^c	Weight of the synaptic connection between the jth and the ith neurons, obtained by correlation rule
$Pr(X)$	Probability of ocurrence of an event X
$\mathbb{R}$	Set of real numbers
$\mathbb{H}$	Set of all hypercomplex numbers
$\mathbb{H}^*$	Set of all non-null hypercomplex numbers
$\mathbf{x}^T$	Transposed of vector $\mathbf{x}$
E	Lyapunov energy function
$ln()$	Natural logarithm
$\mathbb{B}$	Binary set $\{+1, -1\}$
$\mathbb{C}$	Set of complex numbers
$\mathbb{U}$	Set of hyperbolic numbers
$\mathbb{D}$	Set of dual numbers
$\mathbb{Q}$	Set of quaternions
$\mathbb{S}$	Set of unit quaternions or Set of unit complex numbers
$\mathbb{T}$	Set of tessarines
$\mathbb{O}$	Set of octonions
σ	Generalized signum function
$\mathcal{F}$	A family of parametric, continuous, monotone, and increasing functions

$\mathcal{B}$	Symmetric bilinear form
$arg(z)$	Argument of a complex number z
sgn	Generalized signum function
csgn	Complex-valued signum function
$\overline{csgn}$	Complex-valued signum function applying to the natural conjugation to the argument
tsgn	Twin-multistate activation function
κ	Kernel function
$\mathcal{D}$	Subset of a hypercomplex number system $\mathbb{H}$
$\mathcal{S}$	Subset of a hypercomplex number system $\mathbb{H}$
β_i^η	ith component of the vector β^η of Lagrange multipliers

List of Algorithms

Algorithm 1 – Quaternion-valued Recurrent Projection Neural Network 40

Contents

Introduction . . . 18

1 LITERATURE REVIEW . . . 25
1.1 Hopfield Neural Networks . . . 25
1.1.1 Discrete-time Hopfield Neural Networks with Hebbian Learning . . . 25
1.1.2 Hopfield Neural Networks with Projection Rule . . . 28
1.1.3 Complex-Valued Hopfield Neural Networks . . . 28
1.2 Recurrent Correlation Neural Networks . . . 29
1.2.1 Bipolar Recurrent Correlation Neural Networks . . . 29
1.2.2 Complex-Valued Recurrent Correlation Neural Network . . . 32
1.3 Recurrent Kernel Associative Memories . . . 32
1.4 Literature model conclusions . . . 33

2 QUATERNION-VALUED RECURRENT PROJECTION NEURAL NETWORKS . . . 35
2.1 Some Basic Concepts on Quaternions . . . 35
2.2 Quaternion-Valued Models from the Literature . . . 36
2.2.1 Continuous-Valued Quaternionic Hopfield Neural Networks . . . 36
2.2.2 Quaternion-Valued Recurrent Correlation Neural Network . . . 38
2.3 Quaternion-Valued Recurrent Projection Neural Networks on Unit Quaternions . . . 39
2.3.1 Real-valued recurrent projection neural networks (RPNNs) and their relation with other models from the literature . . . 42
2.3.1.1 Optimal storage capacity and generalization of bipolar QRPNNs . . . 42
2.3.1.2 Bipolar RPNNs and their relation with Recurrent Kernel Associative Memories 44
2.3.2 Computational Experiments . . . 46
2.3.2.1 Bipolar Associative Memories . . . 46
2.3.2.2 Quaternion-Valued Associative Memories . . . 48
2.3.2.3 Storage and Recall of Color Images . . . 50
2.4 Conclusions . . . 54

3 HYPERCOMPLEX-VALUED MODELS . . . 55
3.1 Background on Hypercomplex Numbers . . . 55
3.2 A Review of the Hypercomplex-Valued Hopfield Neural Networks 57
3.3 Hypercomplex Recurrent Correlation Neural Networks . . . 59
3.4 Hypercomplex Recurrent Projection Neural Networks . . . 61

3.5 Examples and Computational Experiments 62
3.5.1 HRCNNs synthetic experiments 62
3.5.1.1 Complex-Valued Multistate RCNNs 63
3.5.2 Hyperbolic-Valued Multistate RCNNs 65
3.5.2.1 Quaternion-Valued Multistate RCNNs 68
3.5.2.2 Hypercomplex-valued RCNNs on Cayley-Dickson Algebras 71
3.5.3 HRCNNs Image Experiments 74
3.5.3.1 Storage and Recall of Gray-Scale Images 74
3.5.4 HRPNNs synthetic experiments 76
3.5.5 HRPNNs Image Experiments 79

4 ENSEMBLE OF CLASSIFIERS COMBINED USING RCNNS 83
4.1 Introduction 83
4.2 Ensemble of Binary Classifiers 84
4.2.1 Majority Voting Classifier 85
4.3 Ensemble Based on Bipolar Associative Memories 85
4.3.1 Computational Experiments 89
4.4 Multi-Class Ensemble Classifier 92
4.4.1 Computational Experiments 94

5 CONCLUSIONS 97

Bibliography 99

APPENDIX A- Remarks on the Computational Implementation of HRCNNs 112

Introduction

Artificial neural network models (ANNs) have been developed and studied intensively in the last decades. These computational models, partially based on biological neural networks, can accomplish human-like or super-human like performance in different tasks. Commonly, these models are used in two general classes of problems: classification and regression. This work aims to analyze a specific type of neural network called recurrent neural networks that can implement associative memory models. In order to deepen the study of these networks, we begin with a brief historical review of these models and the biological principles that motivate their study.

In a very simplified way, a neuron is a cell specialized in receiving and transmitting information to another objective cell. The electrical signal coming from another cell enters into the neuron and an electrochemical process called synapse occurs. A synapse can be either excitatory or inhibitory (Haykin, 2009). Transmission lines called axons and terminals called dendrites transmit the electrical signal (Arbib, 1987). This signal is processed and re-transmitted to another neuron(s) or cells when a particular voltage value, called the neuron threshold, is reached.

Motivated by the high degree of complexity in which our brain works and solves problems, the minimal information processing units, called neurons, have been mathematically modeled (Haykin, 2009). The significant progress came from the work of McCulloch and Pitts (1943), which managed to simplify neuronal behavior in five physical hypotheses to create a propositional logic. The study of artificial intelligence begins in parts with the contribution of McCulloch and Pitts (1943).

Mathematically, a weighted sum of the input components, called *the activation potential*, must exceed the threshold value to activate a neuron. This represents the simplification of neuron behavior by McCulloch and Pitts: *The activity of the neuron is an "all-or-none" process* (McCulloch and Pitts, 1943). From this work, the advance in the study of neural networks has been developed. An example is the perceptron of Rosenblatt (1958), taking an important course in the early 80s-90s with the Hopfield (1982) neural network model, and the backpropagation technique (Rumelhart et al., 1986; LeCun et al., 2015). In the 2000s, it became one of the most important topics in science due to the parallel advance in computing power and their many applications in the industry, medicine, and general scientific problems.

In general, it is possible to classify in many different ways the ANNs. One possible description separates them into two broad classes: The feedforward neural networks and the recurrent neural networks. Recurrent networks possess at least one feedback connection, where some outputs are used as a new input for the neuron (Hassoun, 1995).

We are mainly interested in recurrent models, specifically, in the Hopfield model (Hopfield, 1982, 1984), which is one of the main references in this work. The Hopfield neural network (HNN), developed in the early 1980s, is an important and widely-known recurrent neural network which can be used to implement associative memories (AMs) (Hopfield, 1982; Hopfield and Tank, 1985). Successful applications of the Hopfield network include control (Gan, 2017; Song et al., 2017), computer vision and image processing (Wang et al., 2015; Li et al., 2016), classification (Pajares et al., 2010; Zhang et al., 2017), and optimization (Hopfield and Tank, 1985; Serpen, 2008; Li et al., 2015).

A recurrent network defines a sequence of vectors that may eventually converge to a stationary state. The existence of an energy function guarantees that the sequence defined by the recurrent network converges to a local minima of E for any initial state (Hirsh, 1996). This theoretical results may be used to implement an associative memory (AM). Associative memories (AMs) refer to a broad class of mathematical models inspired by the human brain's ability to store and recall information by association (Hassoun and Watta, 1997). The Hopfield (1982) neural network is a typical example of a recurrent neural network able to implement an associative memory. AM models are capable of storing and recalling information. The main idea is to store information in an N-dimensional vectorial form into a system. Then, if we present an incomplete or corrupted version of an element of the memory set as an input, the AM model retrieves the original information. Moreover, the AMs have been extended to fuzzy domains, similarly to correlation recording (Sussner and Valle, 2006; Kosko, 1992).

The recovery process can be implemented using an iterative algorithm to generate a sequence of vectors. In particular, the Hopfield model implementing AMs can be affected by spurious memories (Hopfield and Tank, 1986; Kanter and Sompolinsky, 1987; Personnaz et al., 1985). Briefly, a spurious memory is a stable stationary state that does not belong to the fundamental memory set. In this sense, the Hopfield model suffers from a low storage capacity when it is used to implement an AM. Precisely, due to the cross-talk between the stored items, the Hebbian learning adopted by Hopfield in his original work allows for the storage of approximately $N/(2\ln N)$ items, where N denotes the length of the stored vectors (McEliece et al., 1987). This issue motivated the development of several new models and learning rules in the literature to increase the storage capacity and the noise tolerance of the original bipolar Hopfield network. Firstly, Personnaz et al. (1985) as well as Kanter and Sompolinsky (1987) proposed the projection rule to determine the synaptic weights of the Hopfield networks. The projection rule increases the storage capacity of the Hopfield network to $N-1$ items. Another simple but effective improvement on the storage capacity of the original Hopfield networks was achieved by Chiueh and Goodman (1991, 1993) with the recurrent correlation neural networks (RCNNs). Briefly, an RCNN is obtained by decomposing the Hopfield network with Hebbian learning into a two-layer recurrent neural network. Alternatively, certain RCNNs can be viewed as kernelized versions of the Hopfield network with Hebbian learning (Garcia and Moreno, 2004; Perfetti and Ricci, 2008). The RCNNs as well as the projection rule are part of the core elements for the new models proposed in this thesis: the recurrent projection neural networks (RPNNs).

It comes to light that the associative memory models described in the previous paragraphs are all designed for the storage and recall of bipolar real-valued vectors. However, in many applications, we have to process multivalued or multidimensional data (Hirose, 2012). In view of this remark, the Hopfield neural network as well as the RCNNs have been extended to hypercomplex systems such as complex numbers and quaternions.

Research on complex-valued Hopfield neural networks dates to the late 1980s (Noest, 1988a,b; Aizenberg and Aizenberg, 1992). In 1996, Jankowski et al. (1996) proposed a multistate complex-valued Hopfield network with Hebbian learning that allowed to the development of many other hypercomplex-valued networks. For instance, Lee (2006) developed the projection rule for (multistate) complex-valued Hopfield networks. Based on the works of Jankowski et al. and Lee, Isokawa et al. (2013) proposed a multistate quaternion-valued Hopfield neural network using either Hebbian learning or projection rule. In 2014, Valle (2014a) proposed a complex-valued version of the RCNNs. Recently, the RCNNs have been further extended to quaternions (Valle, 2018).

A revealing behavior of hypercomplex-valued neural networks is that, interpreted as dynamical systems, they are less susceptible to chaotic behavior than their corresponding real-valued versions (de Castro and Valle, 2020). For example, hypercomplex-valued Hopfield networks usually require fewer updates to settle down at an equilibrium state than their corresponding real-valued neural networks. Moreover, hypercomplex-valued versions of neural networks combined with optimization techniques has been demonstrated useful learning representations of multidimensional inputs, obtaining state-of-art results in automatic speech recognition tasks (Ravanelli et al., 2021). Because of this remark, we shall focus on hypercomplex-valued recurrent neural networks that can be used to implement associative memories, proving their stability for a broad class of hypercomplex numbers.

Although real, complex and quaternion-valued RCNNs can be used to implement high-capacity associative memories, they require a sufficiently large parameter which can be impossible in practical implementations (Chiueh and Goodman, 1993; Valle, 2018). In this thesis we circumvent this problem proposing a new model that combines the projection rule and the RCNNs to obtain the new recurrent projection neural networks (RPNNs). We present the quaternion valued recurrent projection neural networks (QRPNNs). As we will show, QRPNNs always have optimal absolute storage capacity. Moreover, the noise tolerance of QRPNNs are usually higher than their corresponding QRCNNs. Besides, we extend the RCNNs to a broad class of hypercomplex number system yielding the so-called hypercomplex-valued RCNNs (HRCNNs). The necessary conditions to ensure that the sequences produced by a HRCNN converge, in both synchronous or asynchronous update mode, is given. Besides, we examine the extension of the RPNN models for other hypercomplex algebras (HRPNNs) such as hyperbolic numbers and octonions. Finally, providing a bridge between recurrent correlation neural networks and ensemble of classifiers, we present several computational experiments in which the HRCNNs are used to combine base classifiers in an ensemble classifier. In particular, the exponential RCNN outperformed several tradicional ensembles of binary classifiers.

Thesis Organization

This thesis contains five chapters. Chapter 1 is a brief review of associative memory models, in particular, the Hopfield neural network in real and complex domains, the recurrent correlation neural network, and recurrent kernel neural networks.

In chapter 2, we introduce the quaternionic recurrent projection neural networks on unit quaternions. Precisely, Chapter 2 includes:

- A brief review on quaternion-valued Hopfield neural networks and quaternion-valued recurrent correlation neural networks (QRCNNs).
- The matrix-based formulation of QRCNNs and QRPNNs.
- A detailed algorithm describing the new QRPNN, which can also be used to implement QRCNNs.
- Formal results concerning the storage capacity of QRPNNs and their relationship with HNN with projection rule and recurrent kernel associative memories (RKAMs) in the bipolar case.
- Computational experiments, including experiments concerning the storage and recall of color images from the CIFAR dataset.

Chapter 3, explores the basic concepts on hypercomplex numbers. Here, we review the hypercomplex-valued discrete-time Hopfield neural networks and introduce a broad class of hypercomplex-valued RCNNs which include the complex-valued and quaternion-valued RCNN models as particular instances, called hypercomplex-valued recurrent correlation neural network (HRCNNs). Moreover, we provide the necessary conditions which ensure that a hypercomplex-valued RCNN always settles at an equilibrium using either synchronous or asynchronous update mode. Finally, computational experiments confirm the potential application of hypercomplex-valued RCNNs as associative memories designed for the storage and recall of gray-scale images. Also, in Chapter 3, we study the extension of the QRPNNs to a broad class of hypercomplex number systems. Using multi-state activation functions to treat complex, hyperbolic and quaternion numbers, we implement computational experiments to compare the dynamics of the RPNN models and their corresponding RCNN models. Moreover, we design associative memories for the storage and recall of gray-scale images.

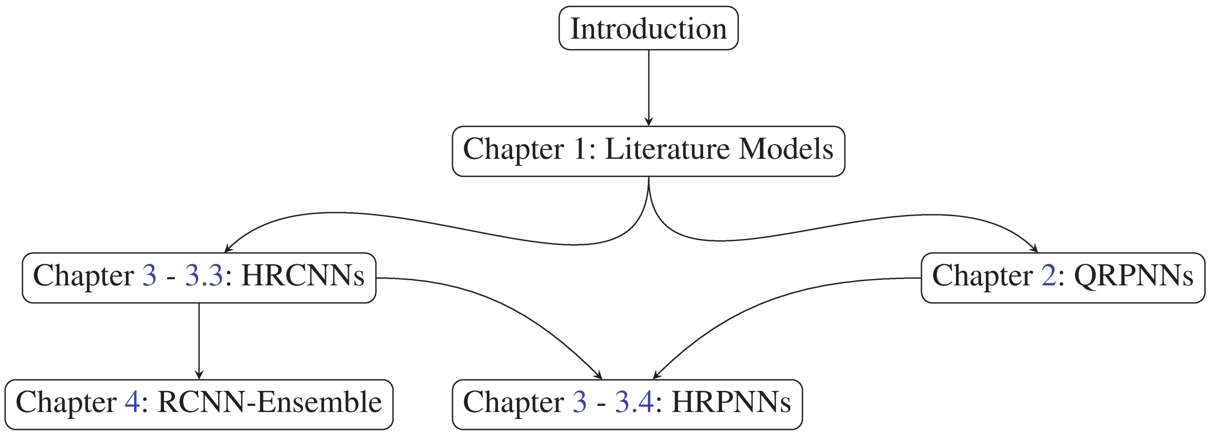


Figure 1 – Diagram that outlines the organization of the thesis.

In chapter 4, we present a new meta-algorithm using the RCNNs as a voting step for classification problems, providing a bridge between RCNNs and ensemble classifiers. Finally, in chapter 5, we present the conclusions and the future works.

We present to the reader the diagram in Figure 1, in order to explain the chapter organization. In particular, HRPNNs is a generalized version of the QRPNNs. However, we present first the QRPNNs due to the direct connections with the literature review in Chapter 1 .

Contribution of the thesis

Our contribution are presented in the chapters 2, 3 and 4.

- In Chapter 2 we introduce the quaternion-valued recurrent projection neural networks (QRPNNs) a new type of recurrent neural network obtained by combining the RCNNs with the projection rule, in order to avoid some storage issues and reach a high capacity associative memory model. Furthermore, this new model generalizes, in the bipolar case, the Hopfield neural networks with the projection rule. We implement associative memories to study the noise tolerance and storage capacity of QRPNNs.

- In the first part of Chapter 3, we extend the quaternion-valued correlation neural networks to a broad class of hypercomplex number systems. We prove that the hypercomplex recurrent correlation neural networks (HRCNNs) generate convergent sequences for any initial input vector, when some particular conditions are satisfied. Consequently, we implement associative memory models for the storage of synthetic and real color images.

- In the second part of Chapter 3, we perform some computational experiments extending the QRPNNs to a broad class of hypercomplex numbers. Here we compare the HRCNNs with these extensions, in order to study of noise tolerance and storage capacity for these new models.

- Finally, in Chapter 4, we present an application of the recurrent correlation associative memories as part of a new meta-algorithm, replacing the voting step in an ensemble classifier to solve binary and multiclass classification problems.

In Figure 2, we present a diagram of the main references of this work. They are organized bottom up chronologically. The blue nodes are the main bipolar models, the green nodes are the main hypercomplex models, the pink nodes are theoretical works, and the yellow nodes are our contributions, including the ensemble classifier application. Finally, the purple node is the extension to hypercomplex numbers of the QRPNN model that we are studying. The edges (arrows) can be interpreted as citation links between the different authors.

As a consequence of this work, we published two conference papers in proceedings and two papers in journals. The author presented his work in the Brazilian Conference on Intelligent Systems (BRACIS2019) (Valle and Lobo, 2019) and (BRACIS2020) (Lobo and Valle, 2020)[1]. Also, the author presented the work in a seminary given by Universidad Tecnológica Metropolitana de Santiago, Chile, called *Seminario Mat-Bio UTEM* for the biomathematical resarch group and in the *Encontro científico dos pós-graduandos do IMECC* (EncPos 2020). The work on QRPNNs has been published in the Theoretical Computer Science journal (Valle and Lobo, 2020). The second paper (Valle and Lobo, 2021) associated to the content of chapter 3, has been published in the Neurocomputing journal. Finally, the models in chapter 3 are implemented in `Julia language` and can be found in open source repositories (see appendix 5).

We would like to point out that, as another contributions within the applied mathematics program, the author collaborated with Dr. Castro and Dr. Valle on incremental learning rules to Hopfield type associative memories. The contribution should be published as a chapter in a book on associative memories by Gate to Computer Science and Research (GCSR) publisher. Implementation and experiments of this work can be found in github [2].

Besides, exploring biomathematical modeling, the author collaborated with on the group of linear parameter varying systems (LPV) of the UNICAMP School of Electrical and Computer Engineering in (Lobo et al., 2019). Also, the author presented a work in *CNMAC 2018 - XXXVIII* (Morillo et al., 2018) , and collaborated in (Barros et al., 2019) and (Sánchez et al., 2019). Finally, the author was part of the organization group for the XIV EncPos (Encontro de Pos-graduandos do IMECC 2019).

[1] You can find the video presentations and also the codes in `https://fitolobo.github.io/`
[2] `https://github.com/fitolobo/Learning-Rules-to-Hopfield-Type-Associative-Memories`

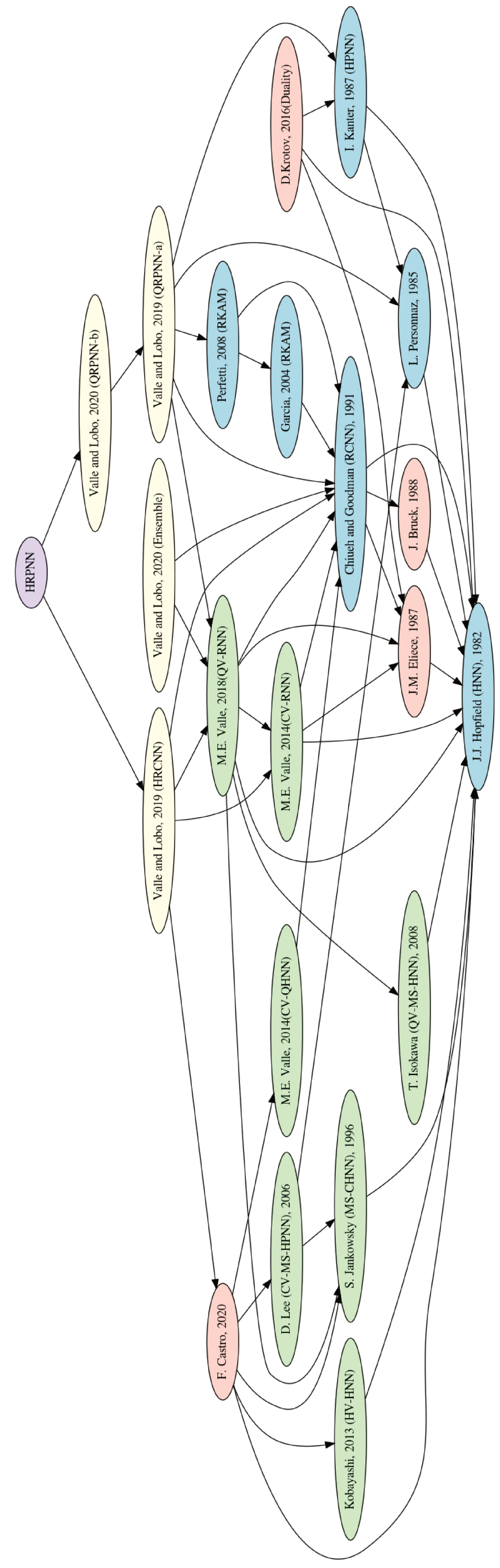

HRPNN
Valle and Lobo, 2020 (QRPNN-b)
Valle and Lobo, 2019 (QRPNN-a)
Valle and Lobo, 2020 (Ensemble)
Valle and Lobo, 2019 (HRCNN)
Perfetti, 2008 (RKAM)
Garcia, 2004 (RKAM)
M.E. Valle, 2018(QV-RNN)
M.E. Valle, 2014(CV-RNN)
M.E. Valle, 2014(CV-QHNN)
D. Lee (CV-MS-HPNN), 2006
F. Castro, 2020
D.Krotov, 2016(Duality)
I. Kanter, 1987 (HPNN)
Chiueh and Goodman (RCNN), 1991
S. Jankowsky (MS-CHNN), 1996
Kobayashi, 2013 (HV-HNN)
L. Personnaz, 1985
J. Bruck, 1988
J.M. Eliece, 1987
T. Isokawa (QV-MS-HNN), 2008
J.J. Hopfield (HNN), 1982

Figure 2 – Main references and the contribution of our work.

1 Literature Review

In this chapter, we present a brief review of the literature. We begin by presenting the Hopfield model and some theoretical aspects of it. In particular, we present the HNN with correlation rule and projection rule. Then we present the complex extension of the HNN model. Moreover, we present the recurrent correlation neural networks (RCNN) in the real-valued and complex-valued cases (Chiueh and Goodman, 1993; Valle, 2014a). Finally, we present the recurrent kernel associative memories. Note that these models have been developed for many years, bringing with them a large number of analysis techniques (Goles-Chacc et al., 1985; McEliece et al., 1987; Bruck, 1990). Moreover, the binary models (Kosko, 1988; Chiueh and Goodman, 1993; Personnaz et al., 1985) have become structurally the basis of new models nowadays (Valle, 2014a; Castro and Valle, 2017b; Isokawa et al., 2012).

1.1 Hopfield Neural Networks

1.1.1 Discrete-time Hopfield Neural Networks with Hebbian Learning

The famous discrete-time Hopfield neural network (HNN) can be thought as a single-layer recurrent neural network. Consider a neural network with N neurons and denote the state of the jth neuron at time t by $x_j(t) \in \mathbb{B} = \{-1, +1\}$. Given the synaptic weights $w_{ij} \in \mathbb{R}$ and an initial state $\mathbf{x}(0)$, the HNN model generates a sequence of vectors by using the activation potential

$$a_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t), \tag{1.1}$$

and the update rule

$$x_j(t+1) = \begin{cases} \sigma\left(a_j(t)\right), & 0 < |a_j(t)|, \\ x_j(t), & \text{otherwise}, \end{cases} \tag{1.2}$$

where σ is the signum function for real numbers.

The topology of the HNN is shown in Figure 3. In order to explain in details the operations involved, the unique layer of the HNN model is represented by using three colored layers. The green circles are the input, the blue hidden circles represents the activation potential given by (1.2) and, finally, the red circles represents the signum function applied in an entry-wise manner.

The sequence defined by the HNNs can be generated by updating one neuron state for iteration, called asynchronous update mode, or updating all neuron states in each iteration, called synchronous update mode. The sequence produced by (1.2) and (1.1) in an asynchronous update

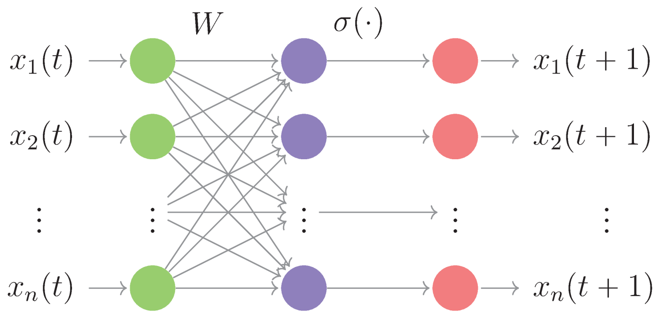


Figure 3 – Hopfield neural network diagram.

mode is convergent for any initial state $\mathbf{x}(0) \in \mathbb{B}^n$ if the synaptic weights satisfies (Hopfield, 1982):

$$w_{ij} = w_{ji} \qquad w_{ii} \geqslant 0, \quad \forall i,j \in \{1,\ldots,N\}. \tag{1.3}$$

Here, the inequality $w_{ii} \geqslant 0$ means that w_{ii} is a non-negative real number.

The convergence of the sequence generated by the HNN is crucial to implement associative memories and other applications. For such purposes, the existence of a Lyapunov energy function is a sufficient condition commonly used to prove the convergence of the sequence generated by recurrent neural network models (Chiueh and Goodman, 1991; Valle, 2014a; Castro and Valle, 2017b). An energy function must satisfy the following conditions:

1. It is a real-valued function on the set of the states of the network.
2. It is bounded from below.
3. The inequality $E(\mathbf{x}(t+\Delta t)) < E(\mathbf{x}(t))$ is always true when $\mathbf{x}(t+\Delta t) \neq \mathbf{x}(t)$.

The existence of an energy function for the recurrent neural network proves that the model yields a convergent sequence for any initial state vector. Depending on the initial state $\mathbf{x}(0)$ the sequence will be attracted to a fixed point of the model. The discrete HNN model has the energy function given by

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} x_i w_{ij} x_j \quad \forall \mathbf{x} \in \mathbb{B}^N. \tag{1.4}$$

The stability analysis allow us to apply the HNN model, for example, as an associative memory (AM). Precisely, if the HNN model always converges to a fixed point for any initial state $\mathbf{x}(0)$, the network yields a mapping $\mathcal{M}$ defined by

$$\mathcal{M}(\mathbf{x}) = \lim_{t\to\infty} \mathbf{x}(t), \tag{1.5}$$

which can be used to implement an associative memory as follows: given the set of vectors $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, called the set of fundamental memories, the map $\mathcal{M}$ is expected to satisfy

$\mathcal{M}(\mathbf{u}^\xi) = \mathbf{u}^\xi$ for all $\xi = 1, \ldots, P$. Moreover it should exhibit some noise tolerance, that is, given a corrupted version $\tilde{\mathbf{u}}^\xi$ of a fundamental memory key $\mathbf{u}^\xi$ then $\mathcal{M}(\tilde{\mathbf{u}}^\xi) = \mathbf{u}^\xi$ (Hassoun and Watta, 1997; Haykin, 2009). We would like to point out that the previous statements refer to the so-called auto-associative memories, the main application studied in this thesis. Apart from the auto-associative, we also have the hetero-associative memories in which the input can differs from the output (Hassoun and Watta, 1997; Kosko, 1988)

The *correlation rule*, also called *Hebbian learning,* is a local and incremental recording recipe (Hassoun and Watta, 1997; Hebb, 1949) which has been adopted by Hopfield in his seminal work for auto-associative memories (Hopfield, 1982). Given a set of real-valued vectors $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, where $\mathbf{u}^\xi \in \mathbb{B}^N$, the synaptic weights by using the *Hebb* rule are given by

$$w^c_{ij} = \frac{1}{N}\sum_{\xi=1}^{P} u^\xi_i u^\xi_j, \quad \forall i, j \in \{1, 2, \ldots, N\}. \tag{1.6}$$

Note that the synaptic weights given by (1.6) satisfy (1.3). Let us explain one issue associated to the correlation rule using matrix notation. The equation (1.6) can be alternatively represented by the sum of P matrices

$$W^\xi = \mathbf{u}^\xi(\mathbf{u}^\xi)^T \quad \xi = 1, \ldots, P. \tag{1.7}$$

Explicitly, W is given by

$$W = \frac{1}{N}\left(W^1 + W^2 + \cdots + W^P\right). \tag{1.8}$$

Also, the potential activation of the HNN with correlation rule in matrix notation is

$$\mathbf{a} = W\mathbf{x}. \tag{1.9}$$

Thus, if we give as input a vector from the fundamental memory set, for example $\mathbf{u}^\gamma$ for $\gamma \in \{1, \ldots, P\}$, then, replacing W using (1.9) and (1.7) in (1.8), we obtain

$$\begin{aligned}
\mathbf{a} &= \frac{1}{N}\left(W^1 + W^2 + \cdots + W^P\right)\mathbf{u}^\gamma && (1.10)\\
&= \frac{1}{N}\left(\mathbf{u}^1(\mathbf{u}^1)^T + \mathbf{u}^2(\mathbf{u}^2)^T + \cdots + \mathbf{u}^P(\mathbf{u}^P)^T\right)\mathbf{u}^\gamma && (1.11)\\
&= \mathbf{u}^\gamma + \frac{1}{N}\sum_{\substack{\xi=1\\ \xi\neq\gamma}}^{P}\left(\left(u^\xi\right)^T u^\gamma\right)u^\xi. && (1.12)
\end{aligned}$$

In (1.12), we obtain as activation potential a vector which is the fundamental memory that we want to recover plus an additional perturbation called *cross-talk* term. The network produces the desire output when the cross-talk term is close to zero. Randomly selected vectors nearly orthogonals have less contribution to the cross-talk term. However, in the case of linear dependence, the cross-talk term becomes large, causing the appearance of spurious memories (Rojas, 1996).

1.1.2 Hopfield Neural Networks with Projection Rule

The *projection rule*, also known as the *generalized-inverse recording recipe,* is another recording recipe which can be used in the storage phase of HNNs. Unlike the HNN with Hebbian learning, the *Hopfield neural network with projection rule* uses a non-local storage prescription that can suppress the cross-talk effect between $\mathbf{u}^1, \ldots, \mathbf{u}^P$ (Kanter and Sompolinsky, 1987). This model is constructed using linear algebra tools, maintaining the updating mode and the σ function given by (1.2). The aim is to find a better storage matrix $W \in \mathbb{R}^{N\times N}$ that is defined as the solution of the minimization problem

$$\min_{W\in\mathbb{R}^{N\times N}} \sum_{\xi=1}^{P} ||W\mathbf{u}^\xi - \mathbf{u}^\xi||_2^2. \tag{1.13}$$

Formally, given the matrix $U \in \mathbb{B}^{N\times P}$, where each column is a fundamental memory element. The solution of (1.13) yields the synaptic weights (Golub and van Loan, 1996)

$$w_{ij}^p = (UU^\dagger)_{ij}, \quad \forall i, j = 1, \ldots, N, \tag{1.14}$$

where $U^\dagger \in \mathbb{R}^{P\times N}$ denotes the pseudoinverse matrix of U. Changing the weight matrix in (1.1) by (1.14), we obtain the HNN with projection rule. It is not hard to show that, if the memory elements are linearly independent, then $\sum_{j=1}^{N} w_{ij}^p u_j^\xi = u_i^\xi$ for all $\xi = 1, \ldots, P$ and $i = 1, \ldots, N$ (Personnaz et al., 1985). Thus, all memory elements are fixed points for the HNN with the projection rule. If the matrix $C = U^T U$ is invertible, the w_{ij}^p elements are alternatively given by the equation

$$w_{ij}^p = \frac{1}{N}\sum_{\eta=1}^{P}\sum_{\xi=1}^{P} u_i^\eta c_{\eta\xi}^{-1} u_j^\xi, \quad \forall i, j \in \{1, \ldots, N\} \tag{1.15}$$

Accordingly, we define $c_{\eta\xi}^{-1}$ as the (η, ξ)-entry of the inverse of the real-valued matrix $C \in \mathbb{R}^{P\times P}$ given by

$$c_{\eta\xi} = \frac{1}{N}\left\langle \mathbf{u}^\xi, \mathbf{u}^\eta \right\rangle, \quad \forall \eta, \xi \in \{1, \ldots, P\}. \tag{1.16}$$

On the downside, the projection rule requires the calculation of $U^\dagger$ or C^{-1}. Thus, it is computationally more expensive than the correlation rule.

1.1.3 Complex-Valued Hopfield Neural Networks

Complex-valued Hopfield neural networks (CvHNNs) are the extension of the real-valued HNN into a complex domain. Let $\mathbb{S}$ denote the set of states of a complex-valued neuron and $\mathcal{D} \subset \mathbb{C}$ denotes the domain of the activation function $\phi : \mathcal{D} \to \mathbb{S}$. The construction of the weight matrix is usually analogous to the real case. Given $\mathcal{U} = \{\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^P\} \subseteq \mathbb{S}^N$, we can store these elements by using the Hebbian learning or the projection rule (Noest, 1988a,b; Hirose, 2012; Lee, 2006). Given an input vector $\mathbf{z}(0) \in \mathbb{S}^N$, we can define recursively a sequence of complex-valued vectors by

$$z_j(t+1) = \begin{cases} \phi\left(a_j(t)\right), & a_j(t) \in \mathcal{D}, \\ z_j(t), & \text{otherwise}, \end{cases} \tag{1.17}$$

where

$$a_i(t) = \sum_{j=1}^{N} w_{ij} z_j(t). \tag{1.18}$$

In relation to the HNN, the main difference is the activation function ϕ which is a complex-valued activation function with domain and co-domain in $\mathbb{C}$. Using the complex-valued multistate signum function of Aizenberg and Aizenberg (1992), Jankowski et al. (1996) proposed a complex-valued Hopfield network which allowed to the development of many other hypercomplex-valued recurrent neural networks including (Lee, 2006; Müezzinoğlu et al., 2003; Tanaka and Aihara, 2009; Kobayashi, 2017c,d; Castro and Valle, 2018; Isokawa et al., 2018). The complex-valued signum function, as described by de Castro and Valle (2020) and Kobayashi (2017c,d) is defined as follows: Given a positive integer $K > 1$, referred to as the *resolution factor*, define $\Delta\theta = \pi/K$ and the sets

$$\mathcal{D} = \{z \in \mathbb{C}\backslash\{0\} : \arg(z) \neq (2k-1)\Delta\theta, \forall k = 1, \ldots, K\}, \tag{1.19}$$

and

$$\mathbb{S} = \{1, e^{2\mathbf{i}\Delta\theta}, e^{4\mathbf{i}\Delta\theta}, \ldots, e^{2(K-1)\mathbf{i}\Delta\theta}\}. \tag{1.20}$$

Then, the complex-signum function $\operatorname{csgn} : \mathcal{D} \to \mathbb{S}$ is given by the following expression for all $z \in \mathcal{D}$:

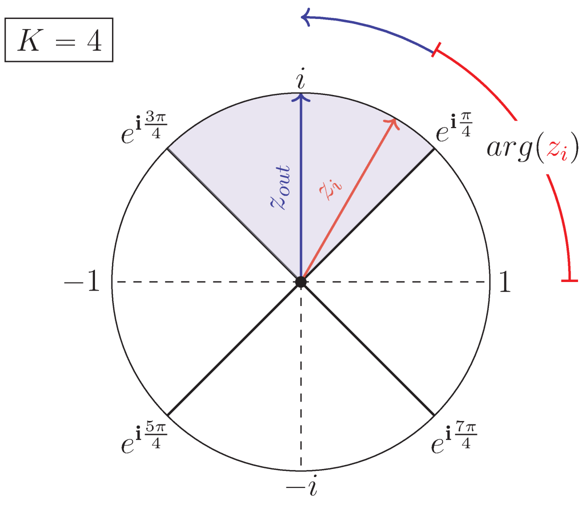
$$\operatorname{csgn}(z) = \begin{cases} 1, & 0 \leqslant \arg(z) < \Delta\theta, \\ e^{2\mathbf{i}\Delta\theta}, & \Delta\theta < \arg(z) < 3\Delta\theta, \\ \vdots & \vdots \\ 1, & (2K-1)\Delta\theta < \arg(z) < 2\pi. \end{cases} \tag{1.21}$$

Figure 4 illustrates the csgn function for a resolution factor $K = 4$. Note that the unit circle is divided in four circular sectors. Given a complex number z_i, we calculate $arg(z_i)$. Then, by using (1.21), we obtain an element of $\mathbb{S}$ given by (1.20).

1.2 Recurrent Correlation Neural Networks

1.2.1 Bipolar Recurrent Correlation Neural Networks

Recurrent correlation neural networks (RCNNs), formerly known as recurrent correlation associative memories (RCAMs), have been introduced by Chiueh and Goodman (1991)



$K = 4$
i
$e^{\mathbf{i}\frac{3\pi}{4}}$
$e^{\mathbf{i}\frac{\pi}{4}}$
$arg(z_i)$
z_{out}
z_i
-1
1
$e^{\mathbf{i}\frac{5\pi}{4}}$
$e^{\mathbf{i}\frac{7\pi}{4}}$
$-i$

Figure 4 – Geometrical example for the csgn function with $K = 4$.

for the storage and recall of N-bit vectors. Precisely, in contrast to the original Hopfield neural network which has a limited storage capacity, some RCNN models can reach the storage capacity of an ideal associative memory (Hassoun and Youssef, 1988). In order to pave the way for the development of the generalized models introduced in the next chapters, let us derive the RCNNs from the correlation-based HNN described by (1.6), (1.1), and (1.2) .

Formally, consider a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \subset \mathbb{B}^N$. Using the synaptic weights w_{ij}^c given by (1.6), we conclude from (1.1) that the activation potential of the ith neuron at iteration t of the correlation-based HNN satisfies

$$\begin{aligned} a_i(t) = \sum_{j=1}^{N} w_{ij}^c x_j(t) &= \sum_{j=1}^{N} \left[\frac{1}{N} \sum_{\xi=1}^{P} u_i^\xi u_j^\xi \right] x_j(t) \\ &= \sum_{\xi=1}^{P} u_i^\xi \left[\frac{1}{N} \sum_{j=1}^{N} u_j^\xi x_j(t) \right] \\ &= \sum_{\xi=1}^{P} u_i^\xi \left[\frac{1}{N} \left\langle \mathbf{x}(t), \mathbf{u}^\xi \right\rangle \right]. \end{aligned}$$

In words, the activation potential $a_i(t)$ is given by a weighted sum of $u_i^1, \ldots, u_i^P$. Moreover, the weights are proportional to the inner product between the current state $\mathbf{x}(t)$ and the fundamental memory $\mathbf{u}^\xi$.

In the RCNN, the activation potential $a_i(t)$ is also given by a weighted sum of $u_i^1, \ldots, u_i^P$. The weights, however, are given by a function of the inner product $\left\langle \mathbf{x}(t), \mathbf{u}^\xi \right\rangle$. Precisely, let $f : [-1, 1] \to \mathbb{R}$ be a (real-valued) continuous and monotone non-decreasing function. Analogously to the HNN, the activation potential of the ith output neuron at time t is defined by

$$a_i(t) = \sum_{\xi=1}^{P} w_\xi(t) u_i^\xi, \quad \forall i = 1, \ldots, N, \tag{1.22}$$

with

$$w_\xi(t) = f\left(\frac{1}{N}\left\langle \mathbf{x}(t), \mathbf{u}^\xi \right\rangle\right), \quad \forall \xi \in 1, \dots, P. \tag{1.23}$$

In short, the first layer of the RCNN computes the inner product (correlation) between the input and the memorized items followed by the evaluation of a non-decreasing continuous activation function. The subsequent layer yields a weighted average of the stored items.

Alternatively, the dynamics of a RCNN can be described using a matrix-vector notation. Let $U = [\mathbf{u}^1, \dots, \mathbf{u}^P] \in \mathbb{B}^{N \times P}$ be the matrix whose columns correspond to the fundamental memories, U^T denote the transpose of U. We assume the functions f and σ are applied in a component-wise manner. Given an initial state $\mathbf{x}(0)$, the dynamics of a RCNN is described by the equations

$$\boldsymbol{w}(t) = f\big(U^T \mathbf{x}(t)/N\big). \tag{1.24}$$

and

$$\mathbf{x}(t+1) = \sigma\big(U \boldsymbol{w}(t)\big), \tag{1.25}$$

where σ is the signum function for real numbers. Examples of RCNNs include:

1. The *correlation RCNN*, or *identity RCNN*, obtained by considering in (1.23) the identity function $f_i(x) = x$.
2. The *high-order RCNN*, which is determined by considering in (1.23) the function
$$f_h(x; q) = (1+x)^q, \quad q > 1. \tag{1.26}$$
3. The *potential-function RCNN*, which is obtained by considering in (1.23) the function
$$f_p(x; L) = \frac{1}{(1 - x + \varepsilon_p)^L}, \quad L \geqslant 1, \tag{1.27}$$
where $\varepsilon_p > 0$ is a small valued introduced to avoid a division by zero when $x = 1$.
4. The *exponential RCNN*, which is determined by considering in (1.23) an exponential function
$$f_e(x; \alpha; \beta) = \beta e^{\alpha x}, \quad \alpha > 0 \quad \beta > 0. \tag{1.28}$$
Here β is a normalization factor to modify the range of the function image.

Let us briefly turn our attention to the high-order, potential-function, and the exponential functions. The functions f_h, f_p, and f_e are all exponential with respect to their parameters. Precisely, these three excitation functions are strictly increasing in $[-1, 1]$ and belong to the following family of parametric functions:

$$\begin{aligned}\mathcal{F} = \{f(x;\lambda) : f(x;\lambda) = [A(x)]^\lambda \text{ for } \lambda \geqslant 0 \text{ and a continuous function } A \\ \text{such that } 0 \leqslant A(x_1) < A(x_2) \text{ when } -1 \leqslant x_1 < x_2 \leqslant 1\}.\end{aligned} \tag{1.29}$$

The family $\mathcal{F}$ is particularly important for the implementation of associative memories (Valle, 2018). Precisely, if the excitation function belongs to the family $\mathcal{F}$, then the RCNN can reach the storage capacity of an ideal associative memory by choosing a sufficiently large parameter λ. On the downside, overflow imposes an upper bound on the parameter λ of an excitation function $f \in \mathcal{F}$, which may limit the application of a RCNN as an associative memory. Since there is a vast literature on RCNN models based on the exponential function f_e (Chiueh and Goodman, 1991; Hancock and Pelillo, 1998; Perfetti and Ricci, 2008), in this work we shall focus on exponential RCNNs and their extensions to other hypercomplex number systems, which we also refer to as *exponential correlation neural network* (ECNN).

1.2.2 Complex-Valued Recurrent Correlation Neural Network

The complex-valued recurrent correlation neural network (CvRCNN) (Valle, 2014a) is a generalization in a complex domain of the bipolar RCNNs. The CvRCNNs are obtained by replacing the McCulloch–Pitts neuron by the continuous-valued neuron of Aizenberg et al. (2005). Let $\mathbb{S}$ denote the set of states of a complex-valued neuron given by $\mathbb{S} = \{z \in \mathbb{C} : |z| = 1\}$. The continuous valued activation function is defined by

$$\sigma = \frac{z}{|z|}, \quad \forall z \neq 0 \tag{1.30}$$

Let $\mathcal{U} = \{\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^P\} \subseteq \mathbb{S}^N$ and $f : [-1, 1] \to \mathbb{R}$ be a continuous and monotone, non-decreasing function. Given a complex-valued input vector $\mathbf{z}(0) = [z_1(0), \ldots, z_N(0)]^T \in \mathbb{S}^N$, the CvRCNN defines recursively the following sequence for $t \geqslant 0$:

$$z_j(t+1) = \sigma\left(\sum_{\xi=1}^{P} w_\xi(t) u_j^\xi\right) \quad \forall j = 1, \ldots, N \tag{1.31}$$

with

$$w_\xi(t) = f\left(\frac{1}{N}\text{Re}\left\{\left\langle \boldsymbol{z}(t), \mathbf{u}^\xi \right\rangle\right\}\right), \quad \forall \xi \in 1, \ldots, P. \tag{1.32}$$

where the weights w's can be given by the functions (1.26), (1.27), (1.28) or the identity function applied to the real part of the inner product in (1.32).

The CvRCNN always produces a convergent sequence, for any initial input vector (Valle, 2014a). As a consequence, the CvRCNNs are used to implement AMs. In particular, the storage capacity of the exponential CvRCNN scales exponentially with the length of the stored vectors (Valle, 2014a).

1.3 Recurrent Kernel Associative Memories

Recurrent kernel associative memories (RKAMs) has theoretical connections with our proposal in next chapter as well as the RCNNs. Let us briefly review the RKAMs.

Support vector machines (SVM) are the main tool used to construct the RKAMs. The SVM model were introduced by Cortes and Vapnik (1995) as a learning method for binary classification. The main idea is to find a hyperplane which separates in two classes the training data. Precisely, the objective is to find the maximum distance between the hyperplane and the closest data points. Only a small amount of the total data defines the hyperplane, such data (or vectors) are called *support vectors*. The distance between the support vectors and the hyperplane is called *margin*. The method also allows using a kernel function κ, which maps the input data points into a higher dimensional *feature space* through a nonlinear function (Perfetti and Ricci, 2008). The maximization problem of the margin is obtained solving an optimization problem in dual form (Cristianini et al., 2000). Using the SVM approach, the RKAM proposed by García and Moreno (2004a,b) and further investigated by Perfetti and Ricci (2008) can implement an associative memory model. The RKAM model is defined as follows: Let κ denote an inner-product kernel and $\rho > 0$ be a user-defined parameter. Given a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \subseteq \mathbb{B}^N$, define the Lagrange multipliers vector $\boldsymbol{\beta}_i = [\beta_{i1}, \ldots, \beta_{iP}]$ as the solution of the following quadratic problem for $i = 1, \ldots, N$:

$$\begin{cases} \text{minimize} \quad Q(\boldsymbol{\beta}_i) = \dfrac{1}{2}\displaystyle\sum_{\xi,\eta=1}^{P} \beta_i^\xi \beta_i^\eta u_i^\xi u_i^\eta \kappa(\mathbf{u}^\xi, \mathbf{u}^\eta) - \sum_{\xi=1}^{P} \beta_i^\xi, \\ \text{subject to} \quad 0 \leqslant \beta_i^\xi \leqslant \rho, \ \forall \xi = 1, \ldots, P. \end{cases} \tag{1.33}$$

Given a bipolar initial state $\mathbf{x}(0) \in \mathbb{B}^N$, a RKAM evolves according to the following equation for all $i = 1, \ldots, N$:

$$x_i(t+1) = \operatorname{sgn}\left(\sum_{\xi=1}^{P} \beta_i^\xi u_i^\xi \kappa\big(\mathbf{u}^\xi, \mathbf{x}(t)\big) \right). \tag{1.34}$$

In words, the next state $x_i(t+1)$ of the ith neuron of a RKAM corresponds to the output of a support vector machine (SVM) classifier without the bias term determined using the training set $\mathcal{T}_i = \{(\mathbf{u}^\xi, u_i^\xi) : \xi = 1, \ldots, P\}$. Therefore, the design of a RKAM requires, in some sense, training N independent support vector classifiers. Explicitly, one SVM for each neuron of the RKAM. Furthermore, the user-defined parameter ρ controls the trade-off between the training error and the separation margin (Schölkopf and Smola, 2002). In the associative memory context, the larger the parameter ρ is, the larger the storage capacity of the RKAM. Conversely, some fundamental memories may fail to be stationary states of the RKAM if ρ is small.

1.4 Literature model conclusions

In this first chapter, we have made a brief review of discrete-time recurrent neural networks. All the reviewed models can implement associative memories. We presented real and complex-valued models with the aim of progressively introducing the different extensions to other hypercomplex-number systems. From the aforementioned models and architectures, it is possible to observe at least two essential motivations for the study and continuous improvement of these models: their potential to store and recall information using explainable mathematical processes, and their stability associated to energy functions. From a mathematical point of view, both aspects are of extreme interest, and naturally encourage us to explore extensions of these models based on other hypercomplex-number systems.

In the next chapter, we present two quaternion-valued models. First, we present the quaternion-valued RCNN (QRCNN) which, in some sense, is based on the correlation rule (Valle, 2018). Understanding the construction of the QRCNN, we introduce a new quaternion-valued model called quaternion-valued recurrent projection neural network (QRPNN). The new architecture is based on the projection rule and the quaternion-valued RCNN, and exhibits optimal storage capacity and high noise tolerance.

2 Quaternion-Valued Recurrent Projection Neural Networks

In this chapter, we present the quaternion-valued recurrent projection neural networks (QRPNNs). First, we present some basic concepts on quaternions. Then, we briefly describe two models from literature: the quaternion-valued Hopfield neural network (QHNNs), and the quaternion-valued recurrent correlation neural network (QRCNNs). Provided the aforementioned definitions, it is possible to define the QRPNNs. We provide some theoretical results for QRPNNs. Also we perform some computational examples and compare the QRPNNs with other models from the literature.

2.1 Some Basic Concepts on Quaternions

Quaternions constitute a four-dimensional associative but non-commutative hypercomplex number system. Apart from representing three and four-dimensional data as a single entity, quaternions are particularly useful to describe rotations in the three dimensional space (Arena et al., 1998; Bülow, 1999; Kuiper, 1999). Quaternions are hypercomplex numbers that extend the real and complex numbers systems. A quaternion may be regarded as a 4-tuple of real numbers, i.e., $q = (q_0, q_1, q_2, q_3)$. Alternatively, a quaternion q can be written as follows

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}, \tag{2.1}$$

where $\mathbf{i}, \mathbf{j}$, and $\mathbf{k}$ are imaginary numbers that satisfy the following identities:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{2.2}$$

Note that $1, \mathbf{i}, \mathbf{j}$, and $\mathbf{k}$ form a basis for the set of all quaternions, denoted by $\mathbb{Q}$.

A quaternion $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ can also be written as $q = q_0 + \vec{q}$, where q_0 and $\vec{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ are called respectively the real part and the vector part of q. The real and the vector part of a quaternion q are also denoted by $\text{Re}\{q\} := q_0$ and $\text{Ve}\{q\} := \vec{q}$.

The sum $p+q$ of two quaternions $p = p_0+p_1\mathbf{i}+p_2\mathbf{j}+p_3\mathbf{k}$ and $q = q_0+q_1\mathbf{i}+q_2\mathbf{j}+q_3\mathbf{k}$ is the quaternion obtained by adding their components, that is,

$$p + q = (p_0 + q_0) + (p_1 + q_1)\mathbf{i} + (p_2 + q_2)\mathbf{j} + (p_3 + q_3)\mathbf{k}. \tag{2.3}$$

Furthermore, the product pq of two quaternions $p = p_0 + \vec{p}$ and $q = q_0 + \vec{q}$ is the quaternion given by

$$pq = p_0q_0 - \vec{p} \cdot \vec{q} + p_0\vec{q} + q_0\vec{p} + \vec{p} \times \vec{q}, \tag{2.4}$$

where $\vec{p} \cdot \vec{q}$ and $\vec{p} \times \vec{q}$ denote respectively the scalar and cross products commonly defined in vector algebra. Quaternion algebra are implemented in many programming languages, including MATLAB, GNU Octave, Julia Language, and python. We would like to recall that the product of quaternions is not commutative. Thus, special attention should be given to the order of the terms in the quaternion product.

The natural conjugate and the norm of a quaternion $q = q_0 + \vec{q}$, denoted respectively by $\bar{q}$ and $|q|$, are defined by

$$\bar{q} = q_0 - \vec{q} \quad \text{and} \quad |q| = \sqrt{\bar{q}q} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{2.5}$$

We say that q is a *unit* quaternion if $|q| = 1$. We symbol $\mathbb{S}$ denotes the set of all unit quaternions, i.e., $\mathbb{S} = \{q \in \mathbb{Q} : |q| = 1|\}$. Note that $\mathbb{S}$ can be regarded as an hypersphere in $\mathbb{R}^4$.

Another important definition is the inner product of two quaternion-valued column vectors. Given $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{Q}^n$ and $\mathbf{y} = [y_1, \ldots, y_n]^T \in \mathbb{Q}^n$ the inner product is given by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} \bar{y}_i x_i. \tag{2.6}$$

The quaternion-valued function $\sigma : \mathbb{Q}^* \to \mathbb{S}$ given by

$$\sigma(q) = \frac{q}{|q|}, \tag{2.7}$$

maps the set of non-zero quaternions $\mathbb{Q}^* = \mathbb{Q} \backslash \{0\}$ to the set of all unit quaternions $\mathbb{S}$. The function σ can be interpreted as a generalization of the signal function to unit quaternions. We explicitly have

$$\text{Re}\,\{\langle \mathbf{x}, \mathbf{y} \rangle\} = \sum_{i=1}^{n} (y_{i0}x_{i0} + y_{i1}x_{i1} + y_{i2}x_{i2} + y_{i3}x_{i3}), \tag{2.8}$$

which corresponds to the inner product of the real-valued vectors of length $4n$ obtained by concatenating the 4-tuples $(x_{i0}, x_{i1}, x_{i2}, x_{i3})$ and $(y_{i0}, y_{i1}, y_{i2}, y_{i3})$, for $i = 1, \ldots, n$.

Finally, given a quaternion-valued matrix $U \in \mathbb{Q}^{N \times P}$ the transposed conjugated matrix of U is defined by U^*, where the conjugation is the natural conjugation (2.5) applied in an entry-wise manner.

2.2 Quaternion-Valued Models from the Literature

2.2.1 Continuous-Valued Quaternionic Hopfield Neural Networks

Consider a quaternion-valued activation function $\sigma = q/|q|$, whose output is obtained by normalizing a quaternion (Valle, 2014b; Kobayashi, 2016a). The *continuous-valued quaternionic Hopfield neural network* (CV-QHNN) is defined as follows: Let $w_{ij} \in \mathbb{Q}$ denote the jth quaternionic synaptic weight of the ith neuron of a network with N neurons. Also, let the state of the CV-QHNN at time t be represented by a column quaternion-valued vector $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^T \in \mathbb{S}^N$, that is, the unit quaternion $x_i(t) = x_{i0}(t) + x_{i1}(t)\mathbf{i} + x_{i2}(t)\mathbf{j} + x_{i3}(t)\mathbf{k}$ corresponds to the state of the ith neuron at time t. Given an initial state (or input vector) $\mathbf{x}(0) = [x_1, \ldots, x_N]^T \in \mathbb{S}^N$, the CV-QHNNs defines recursively the sequence of quaternion-valued vectors $\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \ldots$ by means of the equation

$$x_j(t+1) = \begin{cases} \sigma\left(a_j(t)\right), & 0 < |a_j(t)| < +\infty, \\ x_j(t), & \text{otherwise}, \end{cases} \tag{2.9}$$

where

$$a_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t), \tag{2.10}$$

is the activation potential of the ith neuron at iteration t. The CV-QHNN can be implemented and analyzed more easily than multistate quaternionic Hopfield neural network models (Isokawa et al., 2013). Furthermore, as far as we know, it is the unique version of the Hopfield network on unit quaternions that always yields a convergent sequence in the asynchronous update mode under the usual conditions on the synaptic weights (Valle and Castro, 2018).

In analogy with the traditional real-valued bipolar Hopfield network, the sequence produced by (2.9) and (2.10) in an asynchronous update mode is convergent for any initial state $\mathbf{x}(0) \in \mathbb{S}^N$ if the synaptic weights satisfy (Valle, 2014b):

$$w_{ij} = \bar{w}_{ji} \quad \text{and} \quad w_{ii} \geqslant 0, \quad \forall i, j \in \{1, \ldots, N\}. \tag{2.11}$$

Here, the inequality $w_{ii} \geqslant 0$ means that w_{ii} is a non-negative real number. Moreover, the synaptic weights of a QHNN are usually determined using either the correlation or projection rule (Isokawa et al., 2013). Both correlation and projection rule yield synaptic weights that satisfy (2.11).

Consider a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$. Each $\mathbf{u}^\xi = [u_1^\xi, \ldots, u_N^\xi]^T$ is a quaternion-valued column vector whose components $u_i^\xi = u_{i\,0}^\xi + u_{i\,1}^\xi\mathbf{i} + u_{i\,2}^\xi\mathbf{j} + u_{i\,3}^\xi\mathbf{k}$ are unit quaternions. In the quaternionic version of the correlation rule (Isokawa et al., 2013), the synaptic weights are given by

$$w_{ij}^c = \frac{1}{N} \sum_{\xi=1}^{P} u_i^\xi \bar{u}_j^\xi, \quad \forall i, j \in \{1, 2, \ldots, N\}. \tag{2.12}$$

Unfortunately, such as the real-valued correlation recording recipe, the quaternionic correlation rule is subject to the cross-talk between the original vectors $\mathbf{u}^1, \ldots, \mathbf{u}^P$. In contrast, the *projection rule*, analogously to the real case, avoids the cross-talk issue. Formally, in the projection rule the synaptic weights are defined by

$$w_{ij}^p = \frac{1}{N} \sum_{\eta=1}^{P} \sum_{\xi=1}^{P} u_i^\eta c_{\eta\xi}^{-1} \bar{u}_j^\xi, \tag{2.13}$$

where $c^{-1}_{\eta\xi}$ denotes the (η, ξ)-entry of the quaternion-valued inverse of the matrix $C \in \mathbb{Q}^{P\times P}$ given by

$$c_{\eta\xi} = \frac{1}{N}\sum_{j=1}^{N} \bar{u}^{\eta}_j u^{\xi}_j = \frac{1}{N}\left\langle \mathbf{u}^{\xi}, \mathbf{u}^{\eta}\right\rangle, \quad \forall \mu, \nu \in \{1, \ldots, P\}. \tag{2.14}$$

Analogously to the real case, if the vectors $\mathbf{u}^{\eta}$ are linearly independent, then all the fundamental memories are fixed points of the CV-QHNN with the projection rule. Moreover, the projection rule also requires the inversion of a $P \times P$ quaternion-valued matrix.

2.2.2 Quaternion-Valued Recurrent Correlation Neural Network

The RCNNs have been generalized for the storage and recall of complex-valued and quaternion-valued vectors (Valle, 2014a, 2018). In the QRCNN, the activation potential $a_i(t)$ is also given by a weighted sum of $u_i^1, \ldots, u_i^P$. Analogously to the complex-valued case, the weights are given by a function of the real part of the inner product $\left\langle \mathbf{x}(t), \mathbf{u}^{\xi}\right\rangle$. Precisely, let $f : [-1, 1] \to \mathbb{R}$ be a (real-valued) continuous and monotone non-decreasing function. Given a quaternionic input vector $\mathbf{x}(0) = [x_1(0), \ldots, x_N(0)]^T \in \mathbb{S}^N$, a QRCNN defines recursively a sequence $\{\mathbf{x}(t)\}_{t\geqslant 0}$ of quaternion-valued vectors by means of the generalized version of the signum function (2.7), where the activation potential of the ith output neuron at time t is given by

$$a_i(t) = \sum_{\xi=1}^{P} w_{\xi}(t) u_i^{\xi}, \quad \forall i = 1, \ldots, N, \tag{2.15}$$

with the same weight structure given by (1.32) and possible weight functions (1.26), (1.27), (1.28), but in a quaternion-valued domain.

Note that QRCNNs generalize both bipolar and complex-valued RCNNs (Chiueh and Goodman, 1991; Valle, 2014a). Precisely, the bipolar and the complex-valued models are obtained by considering vectors $\mathbf{x} = [x_1, \ldots, x_N]^T \in \mathbb{S}^N$ whose components satisfy respectively $x_j = x_{j_0} + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ and $x_j = x_{j_0} + x_{j_1}\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ for all $j = 1, \ldots, N$. Furthermore, the correlation QRCNN generalizes the traditional bipolar correlation-based Hopfield neural network but it does not generalize the correlation-based CV-QHNN. Indeed, in contrast to the correlation-based CV-QHNN, the correlation QRCNN uses only the real part of the inner product $\left\langle \mathbf{x}(t), \mathbf{u}^{\xi}\right\rangle$.

Finally, we would like to point out that, independently of the initial state $\mathbf{x}(0) \in \mathbb{S}^N$, a QRCNN model always yields a convergent sequence $\{\mathbf{x}(t)\}_{t\geqslant 0}$ (Valle, 2018). Therefore, QRCNNs are potential models to implement associative memories. Moreover, due to the non-linearity of the activation functions of the hidden neurons, the high-order, potential-function, and exponential QRCNNs may overcome the rotational invariance problem found on quaternionic Hopfield neural network (Kobayashi, 2016a). On the downside, such as the correlation-based quaternionic Hopfield network, QRCNNs may suffer from cross-talk between the fundamental memories $\mathbf{u}^1, \ldots, \mathbf{u}^P$. Inspired by the projection rule, the next chapter introduces improved models which overcome the cross-talk problem of the RCNNs and QRCNNs.

2.3 Quaternion-Valued Recurrent Projection Neural Networks on Unit Quaternions

The main motivation for the quaternion-valued recurrent projection neural networks (QRPNNs) is to combine the projection rule and the QRCNN models to yield high capacity associative memories. Specifically, using the synaptic weights w_{ij}^p given by (2.13), the activation potential of the ith neuron at time t of the projection-based QHNN is

$$\begin{aligned} a_i(t) &= \sum_{j=1}^N w_{ij}^p x_j(t) = \sum_{j=1}^N \left[\frac{1}{N} \sum_{\eta=1}^P \sum_{\xi=1}^P u_i^\eta c_{\eta\xi}^{-1} \bar{u}_j^\xi \right] x_j(t) \\ &= \sum_{\eta=1}^P \sum_{\xi=1}^P u_i^\eta c_{\eta\xi}^{-1} \left[\frac{1}{N} \sum_{j=1}^N \bar{u}_j^\xi x_j(t) \right] \\ &= \sum_{\xi=1}^P \left(\sum_{\eta=1}^P u_i^\eta c_{\eta\xi}^{-1} \right) \left[\frac{1}{N} \langle \mathbf{x}(t), \mathbf{u}^\xi \rangle \right]. \end{aligned}$$

In analogy to the QRCNN, we replace the term proportional to the inner product between $\mathbf{x}(t)$ and $\mathbf{u}^\eta$ by the weight $w_\xi(t)$ given by (1.32). Accordingly, we define $c_{\eta\xi}^{-1}$ as the (η, ξ)-entry of the inverse of the real-valued matrix $C \in \mathbb{R}^{P\times P}$ given by

$$c_{\eta\xi} = f\left(\frac{1}{N} \text{Re}\left\{ \langle \mathbf{u}^\xi, \mathbf{u}^\eta \rangle \right\} \right), \quad \forall \eta, \xi \in \{1, \ldots, P\}. \tag{2.16}$$

Furthermore, to simplify the computation, we define

$$v_i^\xi = \sum_{\eta=1}^P u_i^\eta c_{\eta\xi}^{-1}, \tag{2.17}$$

for all $i = 1, \ldots, N$ and $\eta = 1, \ldots, P$. Thus, the activation potential of a QRPNN is given by

$$a_i(t) = \sum_{\xi=1}^P w_\xi(t) v_i^\xi, \quad \forall i = 1, \ldots, N. \tag{2.18}$$

Concluding, given a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^p\} \subset \mathbb{S}^N$, define the $P \times P$ real-valued matrix C by means of (2.16) and compute the quaternion-valued vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^P$ using (2.17). Like the QRCNN, given an input vector $\mathbf{x}(0) \in \mathbb{S}^N$, a QRPNN yields the sequence $\{\mathbf{x}(t)\}_{t\geqslant 0}$ by means of the equation

$$x_j(t+1) = \begin{cases} \sigma\left(a_j(t)\right), & 0 < |a_j(t)| < +\infty, \\ x_j(t), & \text{otherwise}, \end{cases} \tag{2.19}$$

Algorithm 1 – Quaternion-valued Recurrent Projection Neural Network

```
Data:
    1. A continuous and non-decreasing real-valued function $f : \mathbb{R} \to \mathbb{R}$.
    2. Matrices $U = [\mathbf{u}^1, \ldots, \mathbf{u}^P]$ and $V = [\boldsymbol{v}^1, \ldots, \boldsymbol{v}^P]$.
    3. The input vector $\mathbf{x} = [x_1, \ldots, x_N]$.
    4. Maximum number of iterations $t_{\max}$ and a tolerance $\tau > 0$.
Result: Retrieved vector $\mathbf{y}$.
Initialize $t = 0$ and $\Delta = \tau + 1$.
while $t \leqslant t_{\max}$ and $\Delta \geqslant \tau$ do
    1. Compute the weights
        $w_\xi = f\left(\frac{1}{N}\text{Re}\left\{U^*\mathbf{x}\right\}\right).$
    2. Compute the next state
        $\mathbf{y} = \sigma(V\boldsymbol{w}).$
    3. Update respectively $t \leftarrow t + 1$, $\Delta \leftarrow \|\mathbf{y} - \mathbf{x}\|$, and $\mathbf{x} \leftarrow \mathbf{y}$.
```

and $\sigma : \mathbb{Q}^* \to \mathbb{S}$ given by

$$\sigma(q) = \frac{q}{|q|}. \tag{2.20}$$

Alternatively, using a matrix-vector notation, a synchronous QRPNN can be described as follows: Let $U = [\mathbf{u}^1, \ldots, \mathbf{u}^P] \in \mathbb{S}^{N \times P}$ be the quaternion-valued matrix whose columns corresponds to the fundamental memories. Define the real-valued matrix $C \in \mathbb{R}^{P \times P}$ and the quaternion-valued matrix $V \in \mathbb{Q}^{N \times P}$ by means of the equations

$$C = f(\text{Re}\left\{U^*U\right\}/N) \quad \text{and} \quad V = UC^{-1}, \tag{2.21}$$

where the excitation function f is evaluated in an entry-wise manner. Note that the two equations in (2.21) are equivalent to (2.16) and (2.17), respectively. Given the initial state $\mathbf{x}(0)$, analogously to the QRCNN, the QRPNN defines recursively

$$\boldsymbol{w}(t) = f\big(\text{Re}\left\{U^*\mathbf{x}(t)\right\}/N\big), \tag{2.22}$$

and

$$\mathbf{x}(t+1) = \sigma\big(V\boldsymbol{w}(t)\big), \tag{2.23}$$

where $f : [-1, 1] \to \mathbb{R}$ and $\sigma : \mathbb{Q}^* \to \mathbb{S}$ are evaluated in a component-wise manner. Algorithm 1, formulated using matrix notation, summarizes the implementation of a QRPNN using synchronous update. We would like to point out that a QRCNN is obtained by setting $V = U$ in Algorithm 1.

Like the QRCNN, a QRPNN is also implemented by the fully connected two layer neural network with P hidden neurons shown in Figure 5b). The difference between the QRCNN and the QRPNN is the synaptic weight matrix of the output layer. In other words, they differ in the way the real-valued vector $\boldsymbol{w}(t)$ is decoded to yield the next state $\mathbf{x}(t)$.

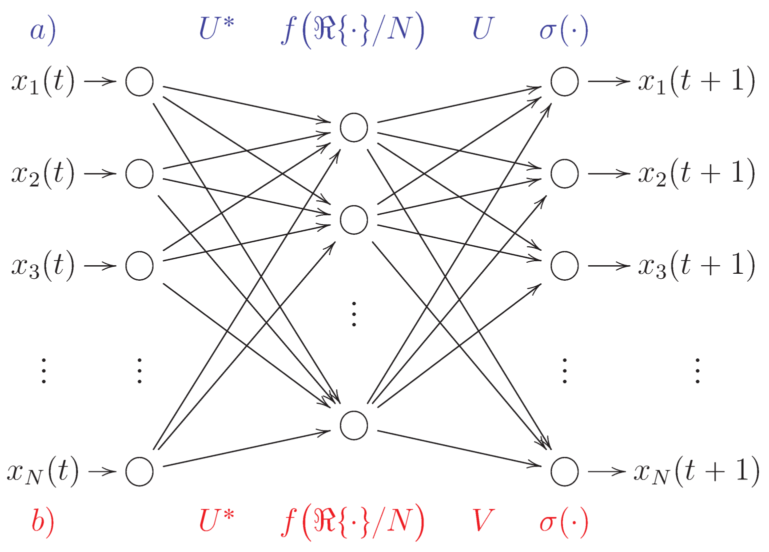


Figure 5 – The network topology of quaternionic recurrent correlation and projection neural networks.

From the computational point of view, although the training phase of a QRPNN requires $\mathcal{O}(P^3 + NP^2)$ operations to compute the matrices C^{-1} and V, they usually exhibit better noise tolerance than the QRCNNs. Moreover, the following theorem shows that QRCNNs overcome the cross-talk between the fundamental memories if the matrix C is invertible.

Example 1. *Consider the fundamental memory matrix $U \in \mathbb{S}^{3\times 2}$, given by:*

$$U = \begin{pmatrix} 0.47796 - 0.56512\mathbf{i} + 0.65827\mathbf{j} + 0.13731\mathbf{k} & -0.48995 + -0.74886\mathbf{i} + 0.05784\mathbf{j} - 0.44248\mathbf{k} \\ 0.08232 + 0.70631\mathbf{i} + 0.65437\mathbf{j} - 0.25716\mathbf{k} & -0.68329 + 0.03334\mathbf{i} - 0.36696\mathbf{j} - 0.63034\mathbf{k} \\ 0.76771 + 0.28354\mathbf{i} - 0.54316\mathbf{j} - 0.18759\mathbf{k} & 0.01761 - 0.20041\mathbf{i} - 0.36471\mathbf{j} + 0.90912\mathbf{k} \end{pmatrix}$$

Then, we calculate C using the exponential function with $\alpha = 2$, obtaining:

$$C = \begin{pmatrix} 7.3890 & 1.0269 \\ 1.0269 & 7.3890 \end{pmatrix}$$

Following Algorithm 1, we calculate the matrix $V = UC^{-1}$ using (2.17),

$$V = \begin{pmatrix} 0.07535 - 0.06362\mathbf{i} + 0.08973\mathbf{j} + 0.02743\mathbf{k} & -0.07678 - 0.09250\mathbf{i} - 0.00464\mathbf{j} - 0.06369\mathbf{k} \\ 0.02446 + 0.0968\mathbf{i} + 0.09734\mathbf{j} - 0.02339\mathbf{k} & -0.09587 - 0.00894\mathbf{i} - 0.06319\mathbf{j} - 0.08205\mathbf{k} \\ 0.10560 + 0.04297\mathbf{i} - 0.06796\mathbf{j} - 0.04332\mathbf{k} & -0.01229 - 0.03309\mathbf{i} - 0.03991\mathbf{j} + 0.12905\mathbf{k} \end{pmatrix}$$

Given the input vector

$$\mathbf{x}(0) = \begin{pmatrix} 0.07987 + 0.55354\mathbf{i} - 0.79031\mathbf{j} - 0.25022\mathbf{k} \\ 0.08232 + 0.70631\mathbf{i} + 0.65437\mathbf{j} - 0.25716\mathbf{k} \\ 0.76771 + 0.28354\mathbf{i} + 0.28354\mathbf{j} - 0.18759\mathbf{k} \end{pmatrix},$$

we obtain the weights

$$\mathbf{w}(0) = \begin{pmatrix} 2.1825714127810736 \\ 0.7093321157481525 \end{pmatrix},$$

and determine

$$\mathbf{x}(1) = \sigma(V\boldsymbol{w}) = \begin{pmatrix} 0.36425 - 0.67707\mathbf{i} + 0.63758\mathbf{j} + 0.04867\mathbf{k} \\ -0.05092 + 0.71466\mathbf{i} + 0.5844\mathbf{j} - 0.38095\mathbf{k} \\ 0.75915 + 0.24070\mathbf{i} - 0.60467\mathbf{j} - 0.01031\mathbf{k} \end{pmatrix}.$$

After seven steps we have

$$\mathbf{x}(7) = \begin{pmatrix} 0.47788 - 0.56521\mathbf{i} + 0.65826\mathbf{j} + 0.13725\mathbf{k} \\ 0.08222 + 0.70633\mathbf{i} + 0.65433\mathbf{j} - 0.25726\mathbf{k} \\ 0.76771 + 0.28351\mathbf{i} - 0.54322\mathbf{j} - 0.18746\mathbf{k} \end{pmatrix}.$$

Note that the distance $d(\mathbf{x}(7), \mathbf{u}^1) = ||\mathbf{x}(7) - \mathbf{u}^1||_2 = 0.0006$. *Indeed, the exponential QRPNN is converging to the first column of matrix* U.

2.3.1 Real-valued recurrent projection neural networks (RPNNs) and their relation with other models from the literature

In the following, we present some theoretical results of the QRPNNs. We also present some relations between the QRPNNs restricted to the real number-system and other models in the literature. We call bipolar QRPNN when the vectors in $\mathcal{U} \in \mathbb{B}^N$ and the input vectors $\mathbf{x} \in \mathbb{B}^N$.

2.3.1.1 Optimal storage capacity and generalization of bipolar QRPNNs

Let us begin by showing that all the fundamental memories are stationary states of a QRPNN if the matrix C is invertible.

Theorem 2.1. *Given a fundamental memory set* $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, *define the real-valued* $P \times P$*-matrix* C *by* (2.16). *If* C *is invertible, then all fundamental memories* $\mathbf{u}^1, \ldots, \mathbf{u}^P$ *are stationary states of a QRPNN defined by* (2.19), (2.18), *and* (2.22).

Proof. Let us assume that the matrix C given by (2.16) is invertible. Also, suppose a QRPNN is initialized at a fundamental memory, that is, $\mathbf{x}(0) = \mathbf{u}^\gamma$ for some $\gamma \in \{1, \ldots, P\}$. From (2.16), and (2.22), we conclude that

$$w_\xi(0) = f\big(\text{Re}\left\{\langle \mathbf{u}^\gamma, \mathbf{u}^\xi \rangle\right\}/N\big) = c_{\xi\gamma}, \quad \forall \xi = 1, \ldots, P.$$

Furthermore, from (2.17) and (2.18), we obtain the following identities for any $i \in \{1, \ldots, N\}$:

$$\begin{aligned} a_i(0) &= \sum_{\xi=1}^P w_\xi(0) v_i^\xi = \sum_{\xi=1}^P \left(\sum_{\eta=1}^P u_i^\eta c_{\eta\xi}^{-1} \right) c_{\xi\gamma} \\ &= \sum_{\eta=1}^P u_i^\eta \left(\sum_{\xi=1}^P c_{\eta\xi}^{-1} c_{\xi\gamma} \right) = \sum_{\eta=1}^P u_i^\eta \delta_{\eta\gamma} = u_i^\gamma, \end{aligned}$$

where $\delta_{\eta\gamma}$ is the Kronecker delta, that is, $\delta_{\eta\gamma} = 1$ if $\eta = \gamma$ and $\delta_{\eta\gamma} = 0$ if $\eta \neq \gamma$. From (2.19), we conclude that the fundamental memory $\mathbf{u}^\gamma$ is a fixed point of the QRPNN if the matrix C is invertible. □

Theorem 2.1 shows that QRPNNs can be used to implement an associative memory whenever the matrix C is invertible. The following theorem shows that the matrix C given by (2.16) is invertible for an excitation function $f \in \mathcal{F}$ with a sufficiently large parameter λ.

Theorem 2.2. *Consider a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \subset \mathbb{S}^N$ and an excitation function $f \in \mathcal{F}$, where $\mathcal{F}$ is the family of parameterized functions given by (1.29). The matrix C given by (2.16) is strictly diagonally dominant and, thus invertible, for a parameter λ sufficiently large.*

Proof. First of all, recall that a matrix C is strictly diagonally dominant if

$$\sum_{\xi \neq \eta} \left| \frac{c_{\eta\xi}}{c_{\eta\eta}} \right| < 1, \quad \forall \eta = 1, \ldots, P. \tag{2.24}$$

Let us assume that C is given by (2.16) for $f \in \mathcal{F}$. In other words, we have

$$c_{\eta\xi} = f\left(\text{Re}\left\{\langle \mathbf{u}^\xi, \mathbf{u}^\eta \rangle\right\}/N; \lambda\right) = \left[A\left(\text{Re}\left\{\langle \mathbf{u}^\xi, \mathbf{u}^\eta \rangle\right\}/N\right)\right]^\lambda > 0. \tag{2.25}$$

By taking the limit $\lambda \to \infty$ in (2.24), we obtain the following identities for $\eta = 1, \ldots, P$:

$$\lim_{\lambda \to \infty} \sum_{\xi \neq \eta} \left| \frac{c_{\eta\xi}}{c_{\eta\eta}} \right| = \lim_{\lambda \to \infty} \sum_{\xi \neq \eta} \left[\frac{A\left(\text{Re}\left\{\langle \mathbf{u}^\xi, \mathbf{u}^\eta \rangle\right\}/N\right)}{A(1)} \right]^\lambda = 0, \tag{2.26}$$

because $\text{Re}\left\{\langle \mathbf{u}^\xi, \mathbf{u}^\eta \rangle\right\}/N < 1$ for $\xi \neq \eta$ and $A(x) < A(1)$ for all $x < 1$. Concluding, there exists a sufficiently large parameter λ such that (2.24) holds true for all $\eta = 1, \ldots, P$. □

Broadly speaking, the proof of Theorem 2.2 shows that C approximates a multiple of the identity matrix as the parameter λ of $f \in \mathcal{F}$ increases. Borrowing the terminology from Perfetti and Ricci (2008), we say that a QRCNN as well as a QRPNN are in saturated mode if C can be approximated by cI, for some $c > 0$. In the saturated mode, QRCNNs and QRPNNs coincide.

In fact, given an arbitrary state vector $\mathbf{x} \in \mathbb{S}^N$, let $\mathbf{x}_P$ and $\mathbf{x}_C$ denote respectively the states of the QRPNN and QRCNN models after one single synchronous update, that is,

$$\mathbf{x}_P = \sigma(V\boldsymbol{w}) \quad \text{and} \quad \mathbf{x}_C = \sigma(U\boldsymbol{w}), \tag{2.27}$$

where $\boldsymbol{w} = f(\text{Re}\{U^*\mathbf{x}\}/N)$. In the saturated mode, C^{-1} can be approximated by $(1/c)I$, for $c > 0$, and the state vector of the QRPNN satisfies

$$\mathbf{x}_P = \sigma(UC^{-1}\boldsymbol{w}) \approx \sigma\left(\frac{1}{c}U\boldsymbol{w}\right) = \sigma(U\boldsymbol{w}) = \mathbf{x}_C. \tag{2.28}$$

Let us investigate further the relationship between QRPNNs in the bipolar case and HPNNs. Precisely, we show that the bipolar QRPNN, called simply recurrent projection neural network (RPNN), is closely related to the HPNN model.

Theorem 2.3. *The bipolar QRPNN model generalizes the traditional bipolar projection-based Hopfield neural network when the excitation function f is the identity function.*

Proof. Replacing $V = UC^{-1}$ in (2.23) we obtain

$$\mathbf{x}(t+1) = \sigma\left(UC^{-1}\mathbf{w}(t)\right). \tag{2.29}$$

Taking f as the identity function, we have $\mathbf{w}(t) = U^T\mathbf{x}(t)$ and, thus,

$$\mathbf{x}(t+1) = \sigma\left(UC^{-1}U^T\mathbf{x}(t)\right). \tag{2.30}$$

Now, (2.30) is equivalent to

$$\mathbf{x}(t+1) = \sigma\left(U(U^*U/N)^{-1}U^T\mathbf{x}(t)\right). \tag{2.31}$$

Simplifying the argument of σ, we obtain

$$\mathbf{x}(t+1) = \sigma\left(UU^{\dagger}\mathbf{x}(t)\right).$$

This implies that the RPNNs generalizes the HNN with projection rule. $\square$

The identity QRPNN, however, does not generalize the projection-based QHNN because the former uses only the real part of the inner product between $\mathbf{x}(t)$ and $\mathbf{u}^{\xi}$. In fact, in contrast to the projection-based QHNN, the design of a QRPNN does not require the inversion of a quarternion-valued matrix but only the inversion of a real-valued matrix.

2.3.1.2 Bipolar RPNNs and their relation with Recurrent Kernel Associative Memories

As pointed out previously, quaternion-valued RPNNs reduce to bipolar models when the fundamental memories are all real-valued, that is, their vector part is zero. In this subsection, we address the relationship between bipolar RPNNs and RKAMs.

Let us compare the RKAMs with bipolar RCNNs and RPNNs. To this end, let us assume the excitation function f is a valid kernel. The exponential function f_e, for example, yields a valid kernel, namely the Gaussian radial-basis function kernel (Perfetti and Ricci, 2008). As pointed out by Perfetti and Ricci (2008), the main difference between a RKAM and a RCNN is the presence of the Lagrange multipliers in the former. Precisely, the Lagrange multipliers are $\beta_i^{\xi} = 1$ in the RCNNs while, in the RKAM, they are obtained solving (1.33). Furthermore, the RKAM is equivalent to the corresponding bipolar RCNN in the saturation mode, that is, when the matrix C given by (2.16) exhibits a diagonal dominance (Perfetti and Ricci, 2008). In a similar fashion, we observe that a RKAM and a RPNN coincide if $v_i^{\xi} = \beta_i^{\xi} u_i^{\xi}$ for all $i = 1, \ldots, N$ and $\xi = 1, \ldots, P$. The following theorem shows that this equation holds true if all the constraints are inactive at the solution of the quadratic problem (1.33).

Theorem 2.4. *Let $f : [-1,+1] \to \mathbb{R}$ be a continuous and non-decreasing function such that the following equation yields a valid kernel*

$$\kappa(\mathbf{x},\mathbf{y}) = f\left(\frac{\langle \mathbf{x},\mathbf{y}\rangle}{N}\right), \quad \forall \mathbf{x},\mathbf{y} \in \{-1,+1\}^N. \tag{2.32}$$

Consider a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1,\ldots,\mathbf{u}^P\} \subset \{-1,+1\}^N$ such that the matrix C given by (2.16) *is invertible. If the solutions of the quadratic problem defined by* (1.33) *satisfy $0 < \beta_i^\xi < \rho$ for all $i = 1,\ldots,N$ and $\xi = 1,\ldots,P$, then the RKAM defined by* (1.33) *and* (1.34) *coincide with the RPNN defined by* (2.22) *and* (2.23)*. Alternatively, the RKAM and the bipolar RPNN coincide if the vectors $\mathbf{v}^\xi$'s given by* (2.17) *satisfy $0 < v_i^\xi u_i^\xi < \rho$ for all $i = 1,\ldots,N$ and $\xi = 1,\ldots,P$.*

Proof. First of all, note that $\kappa(\mathbf{u}^\xi,\mathbf{u}^\eta) = c_{\xi\eta}$ given by (2.16).

Let us first show that the RKAM coincide with the bipolar RPNN if the inequalities $0 < \beta_i^\xi < \rho$ hold true for all i and ξ. If there is no active constrains at the solution of (1.33), then the Lagrange multiplier $\boldsymbol{\beta}_i$ are the solution of the unconstrained quadratic problem (1.33):

$$\text{minimize } Q(\boldsymbol{\beta}_i) = \frac{1}{2}\sum_{\xi,\eta=1}^{P} \beta_i^\xi u_i^\xi c_{\xi\eta} u_i^\eta \beta_i^\eta - \sum_{\xi=1}^{P} \beta_i^\xi, \quad \forall i = 1,\ldots,N. \tag{2.33}$$

It turns out that the minimum of (2.33), obtained by imposing $\dfrac{\partial Q}{\partial \beta_i^\xi} = 0$, is the solution of the linear system

$$\sum_{\eta=1}^{P} u_i^\xi c_{\xi\eta} u_i^\eta \beta_i^\eta = 1, \quad \xi = 1,\ldots,P. \tag{2.34}$$

Multiplying (2.34) by u_i^ξ and recalling that $(u_i^\xi)^2 = 1$, we obtain

$$\sum_{\eta=1}^{P} c_{\xi\eta} u_i^\eta \beta_i^\eta = u_i^\xi, \quad \xi = 1,\ldots,P. \tag{2.35}$$

Since the matrix C is invertible, the solution of the linear system of equations (2.35) is

$$u_i^\xi \beta_i^\xi = \sum_{\eta=1}^{P} c_{\xi\eta}^{-1} u_i^\eta, \quad \forall \xi = 1,\ldots,P \quad \text{and} \quad i = 1,\ldots,N, \tag{2.36}$$

where $c_{\xi\eta}^{-1}$ denotes the (ξ,η)-entry of C^{-1}. We conclude the first part of the proof by noting that the right-hand side of (2.36) and (2.17) coincide. Therefore, $v_i^\xi = \beta_i^\xi u_i^\xi$ for all $i = 1,\ldots,N$ and $\xi = 1,\ldots,P$ and the RKAM coincides with the bipolar QRPNN.

On the other hand, if $0 < v_i^\xi u_i^\xi < \rho$ for all $i = 1,\ldots,N$ and $\xi = 1,\ldots,P$, then $v_i^\xi = \beta_i^\xi u_i^\xi$ is a solution of (2.36). Equivalently, $\beta_i^\xi = u_i^\xi v_i^\xi$ is the solution of the unconstrained quadratic problem (2.33) as well as the quadratic problem with bounded constraints (1.33). As a consequence, the RKAM defined by (1.33) and (1.34) coincides with the RPNN defined by (2.23) using the RCNN evolution equation. $\square$

We would like to point out that the condition $0 < \beta_i^\xi < \rho$ often occurs in the saturation mode, that is, when the matrix C exhibits a diagonal dominance. In particular, the matrix C is diagonally dominant given a sufficiently large parameter λ of an excitation function $f \in \mathcal{F}$, where $\mathcal{F}$ is a family of continuous, non-decreasing functions given by (1.29). In fact, given a parametric function $f \in \mathcal{F}$, the Lagrange multiplier β_i^ξ approaches $1/f(1;\lambda)$ as λ increases (Perfetti and Ricci, 2008). Concluding, in the saturation mode, the RKAM, RCNN, and RPNN are all equivalent. We shall confirm this remark in the computational experiments presented in the next section.

2.3.2 Computational Experiments

This section provides computational experiments comparing the performance of QHNNs, QRCNNs, and the new QRPNN models as associative memory models. Let us begin by addressing the noise tolerance and storage capacity of the recurrent neural networks for the storage and recall of bipolar real-valued vectors. The bipolar case is an interesting case of study because it is used to confirm the theoretical relation given by Theorem 2.4 between RKAMs and RPNNs. We address the noise tolerance and storage capacity of quaternion-valued vectors subsequently.

2.3.2.1 Bipolar Associative Memories

Let us compare the storage capacity and noise tolerance of the Hopfield neural networks (HNNs), the original RCNNs, and the new RPNNs designed for the storage of $P = 36$ randomly generated bipolar (real-valued) vectors of length $N = 100$. Precisely, we consider the correlation-based and projection-based Hopfield neural networks, the identity, high-order, potential-function, and exponential RCNN and RPNN models with parameters $q = 5$, $L = 3$, and $\alpha = 4$, respectively.

To evaluate the storage capacity and noise tolerance of the bipolar associative memories, the following steps have been repeated 100 times for $N = 100$ and $P = 36$:

1. We synthesized associative memories designed for the storage and recall of a randomly generated fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \subset \{-1, +1\}^N$, where Pr$[u_i^\xi = 1] =$ Pr$[u_i^\xi = -1] = 0.5$ for all $i = 1, \ldots, n$ and $\xi = 1, \ldots, P$.
2. We probed the associative memories with an input vector $\mathbf{x}(0) = [x_1(0), \ldots, x_n(0)]^T$ obtained by reversing some components of $\mathbf{u}^1$ with probability π, i.e., Pr$[x_i(0) = -u_i^1] = \pi$ and Pr$[x_i(0) = u_i^1] = 1 - \pi$, for all i and ξ.
3. The associative memories have been iterated until they reached a stationary state or completed a maximum of 1000 iterations. A memory model succeeded to recall a stored item if the output $\mathbf{y}$ satisfies $||\mathbf{u}^1 - \mathbf{y}|| < \tau$ where $\tau = 10^{-4}$. We count the number of

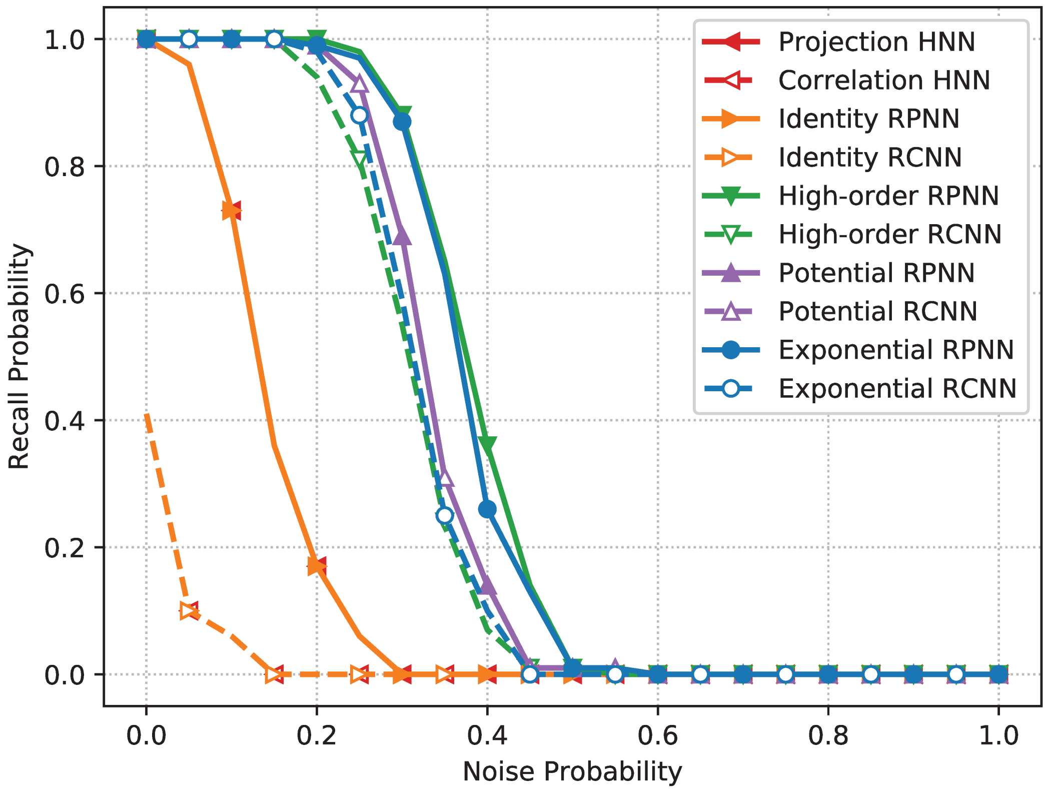


Figure 6 – Recall probability of bipolar associative memories by the noise intensity introduced in the input vector. For each level of noise the experiment is repeated 100 times, and is calculated the probability of recalling a stored item.

succeeded recalls for each noise level and, finally, we divided by the total number of simulations to calculate the recall probability.

Figure 6 shows the probability of an associative memory to recall a fundamental memory by the probability of noise introduced in the initial state. Note that the projection-based HNN coincides with the identity RPNN. Similarly, the correlation-based HNN coincides with the identity RCNN. Also, note that the RPNNs always succeeded to recall undistorted fundamental memories (zero noise probability). The high-order, potential-function, and exponential RCNNs also succeeded to recall undistorted fundamental memories. Nevertheless, the recall probability of the high-order and exponential RPNNs are greater than or equal to the recall probability of the corresponding RCNNs. In other words, the RPNNs exhibit better noise tolerance than the corresponding RCNNs. The potential-function RCNN and RPNN yielded similar recall probabilities.

In a similar fashion, let us compare the storage capacity and noise tolerance of the RCNN, RPNN, and the RKAM with kernel given by (2.32) with $f \equiv f_e$. In this experiment, we considered $\rho = 1000$ and $\alpha = 1$ and $\alpha = 3$. Figure 7 shows the recall probabilities of the

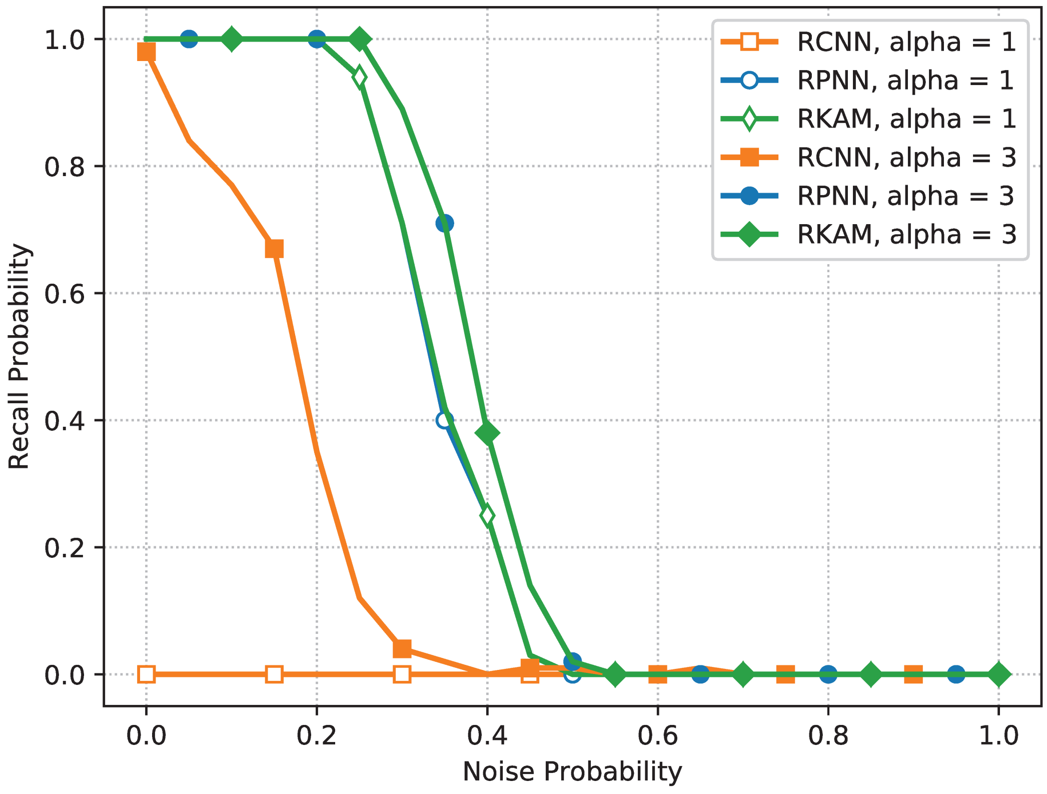


Figure 7 – Recall probability of exponential bipolar RPNN and RKAM by the noise intensity introduced in the input vector.

exponential bipolar RCNN and RKAM, with different parameter values, by the noise probability introduced in the input. Note that both RPNN and RKAM outperformed the RCNN. Furthermore, the exponential bipolar RPNN and the RKAM coincided for all the values of the parameter $\alpha \in \{1, 3\}$. According to Theorem 2.4, an RKAM coincide with a bipolar RPNN if the Lagrange multipliers satisfy $0 < \beta_i^\xi < \rho$ for all $i = 1, \ldots, N$ and $\xi = 1, \ldots, P$. Figure 8 shows the histogram of the Lagrange multipliers obtained solving (1.33) for a random generated matrix U and the exponential kernel with $\alpha = 1, 2$, and 3. The vertical dotted lines correspond to the values e^{-3}, e^{-2}, and e^{-1}. Note that β_i^ξ approaches $1/f(1) = e^{-\alpha}$ as α increases. More importantly, the inequalities $0 < \beta_i^\xi < \rho$ are satisfied for $\alpha > 1$ and $\rho \geqslant 2$.

2.3.2.2 Quaternion-Valued Associative Memories

Let us now investigate the storage capacity and noise tolerance of the asssociative memory models for the storage and recall of $P = 36$ randomly generated quaternion-valued vectors of length $N = 100$. In this example, we considered the projection-based and the correlation-based quaternion-valued Hopfield neural network (QHNNs) as well as the identity, high-order, potential-function, and exponential QRCNNs and QRPNNs with parameters $q = 20$, $L = 3$, and $\alpha = 15$. These parameters have been determined so that the QRCNNs have more

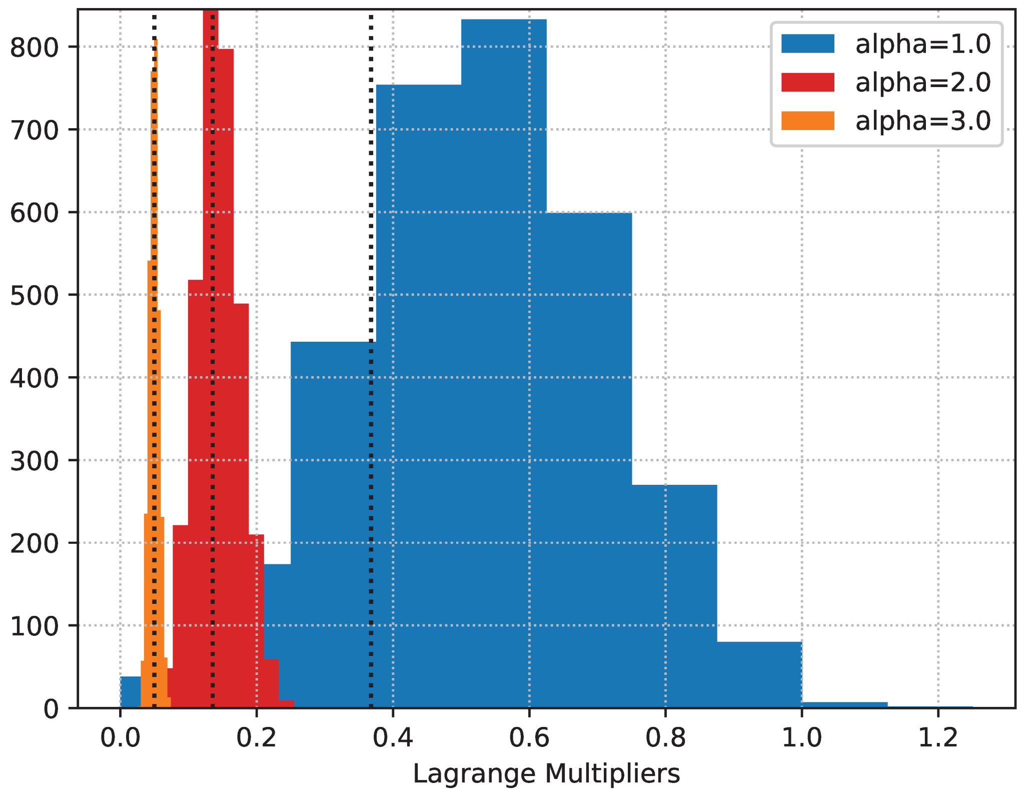


Figure 8 – Histogram of the Lagrange multipliers of a RKAM model.

than 50% probability to recall an undistorted fundamental memory. In analogy to the previous example, the following steps have been performed 100 times:

1. We synthesized associative memories designed for the storage and recall of uniformly distributed fundamental memories $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$. Formally, we defined $u_i^\xi = \texttt{RandQ}$ for all $i = 1, \ldots, N$ and $\xi = 1, \ldots, P$ where

$$\texttt{RandQ} = (\cos\phi + \mathbf{i}\text{sen}\phi)(\cos\psi + \mathbf{k}\text{sen}\psi)(\cos\theta + \mathbf{j}\text{sen}\theta),$$

   is a randomly generated unit quaternion obtained by sampling angles $\phi \in [-\pi, \pi)$, $\psi \in [-\pi/4, \pi/4]$, and $\theta \in [-\pi/2, \pi/2)$ using an uniform distribution.

2. We probed the associative memories with an input vector $\mathbf{x}(0) = [x_1(0), \ldots, x_N(0)]^T$ obtained by replacing some components of $\mathbf{u}^1$ with probability π by an uniformly distributed component, i.e., $\Pr[x_i(0) = \texttt{RandQ}] = \pi$ and $\Pr[x_i(0) = u_i^1] = 1 - \pi$, for all i and ξ.

3. The associative memories have been iterated until they reached a stationary state or completed a maximum of 1000 iterations. The memory model succeeded if the output equals the fundamental memory $\mathbf{u}^1$.

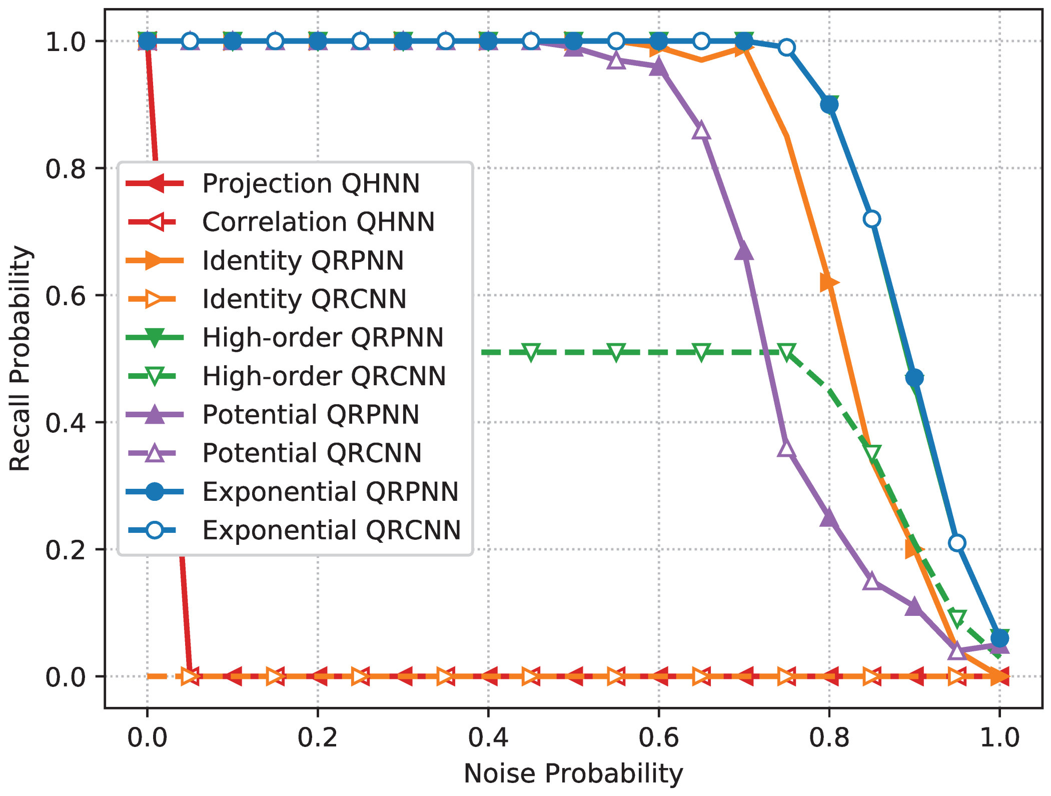


Figure 9 – Recall probability of quaternion-valued associative memories by the noise intensity introduced in the input vector.

Figure 9 shows the probability of a quaternion-valued associative memory to recall a fundamental memory by the probability of noise introduced in the initial state. As expected, the QRPNNs always succeeded to recall undistorted fundamental memories. The potential-function and exponential QRCNNs also succeeded to recall undistorted fundamental memories. Indeed, the potential-function QRCNN and QRPNNs yielded the same recall probability. The noise tolerance of the exponential QRCNN and QRPNN also coincided. Nevertheless, the recall probability of the high-order QRPNN is greater than the recall probability of the corresponding QRCNNs. Furthermore, in contrast to the real-valued case, the projection QHNN differs from the identity QRPNN. In fact, the noise tolerance of the identity QRPNN is far greater than the noise tolerance of the projection QHNN.

2.3.2.3 Storage and Recall of Color Images

In the previous subsection, we compared the performance of QHNN, QRCNN, and QRPNN models designed for the storage and recall of uniformly distributed fundamental memories. Let us now compare the performance of the quaternion-valued associative memories for the storage and recall of color images. Specifically, let us compare the noise tolerance of QHNN, QRCNN, and QRPNN models when the input is a color image corrupted by Gaussian noise. Computationally, an image corrupted by Gaussian noise is obtained by adding a term drawn from an Gaussian distribution with zero mean and a fixed standard variation to each channel of a color image.

At this point, we would like to recall that an RGB color image $\mathbf{I}$ can be converted to a unit quaternion-valued vector $\mathbf{x} = [x_1, \ldots, x_N] \in \mathbb{S}_{\mathbb{Q}}^N$, $N = 1024$, as follows (Castro and Valle, 2017a): Let $I_i^R \in [0,1]$, $I_i^G \in [0,1]$, and $I_i^B \in [0,1]$ denote respectively the red, green, and blue intensities at the ith pixel of an RGB color image $\mathbf{I}$. For $i = 1, \ldots, N$, we first compute the phase-angles

$$\phi_i = (-\pi + \epsilon) + 2(\pi - \epsilon) I_i^R, \tag{2.37}$$

$$\psi_i = \left(-\frac{\pi}{4} + \epsilon\right) + \left(\frac{\pi}{2} - 2\epsilon\right) I_i^G, \tag{2.38}$$

$$\theta_i = \left(-\frac{\pi}{2} + \epsilon\right) + (\pi - 2\epsilon) I_i^B, \tag{2.39}$$

where $\epsilon > 0$ is a small number such that $\phi_i \in [-\pi, \pi)$, $\psi_i \in [-\pi/4, \pi/4]$, and $\theta_i \in [-\pi/2, \pi/2)$ (Bülow, 1999). In our computational experiments, we adopted $\epsilon = 10^{-4}$. Then, we define the unit quaternion-valued vector $\mathbf{x}$ using the phase-angle representation $x_i = e^{\phi_i \mathbf{i}} e^{\psi_i \mathbf{k}} e^{\theta_i \mathbf{j}}$ of its components. Equivalently, we have

$$x_i = (\cos\phi_i \mathbf{i}\text{sen}\phi_i)(\cos\psi_i + \mathbf{k}\text{sen}\psi_i)(\cos\theta_i + \mathbf{j}\text{sen}\theta_i), \quad \forall i = 1, \ldots, N. \tag{2.40}$$

Conversely, given an unit quaternion-valued vector $\mathbf{x}$, we first compute the phase-angles ϕ_i, ψ_i, and θ_i of the component x_i using Table 2.2 in Bülow (1999). Afterwards, we obtain the RGB color image $\mathbf{I}$ by inverting (2.37)-(2.39), that is,

$$I_i^R = \frac{\phi_i + \pi - \epsilon}{2(\pi - \epsilon)}, \quad I_i^G = \frac{\psi_i + \pi/4 - \epsilon}{(\pi/2 - 2\epsilon)}, \quad \text{and} \quad I_i^B = \frac{\theta_i + \pi/2 - \epsilon}{(\pi - 2\epsilon)}, \tag{2.41}$$

for all $i = 1, \ldots, N$.

In order to compare the performance of the quaternion-valued associative memories, we used color images from the CIFAR dataset (Krizhevsky, 2009). Recall that the CIFAR dataset contains 60.000 RGB color images of size 32×32. In this experiment, we randomly selected $P = 200$ color images and converted them to unit quaternion-valued vectors $\mathbf{u}^1, \ldots, \mathbf{u}^P$. Similarly, we corrupted one of the P selected images with Gaussian noise and converted it to a quaternion-valued vector $\mathbf{x} \in \mathbb{S}_{\mathbb{Q}}^N$. The corrupted vector $\mathbf{x}$ have been presented to associative memories designed for the storage of the fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \in \mathbb{S}_{\mathbb{Q}}^N$.

Figure 10 shows an original color image selected from the CIFAR dataset, a color image corrupted by Gaussian noise with standard deviation 0.1, and the corresponding images retrieved by the associative memory models. Note that the correlation-based QHNN as well as the identity QRCNN failed to retrieve the original image due to the cross-talk between the stored items. Although the projection-based QHNN yielded an image visually similar to the original dog's image, this memory model also failed to retrieve the original image due to the cyan pixels

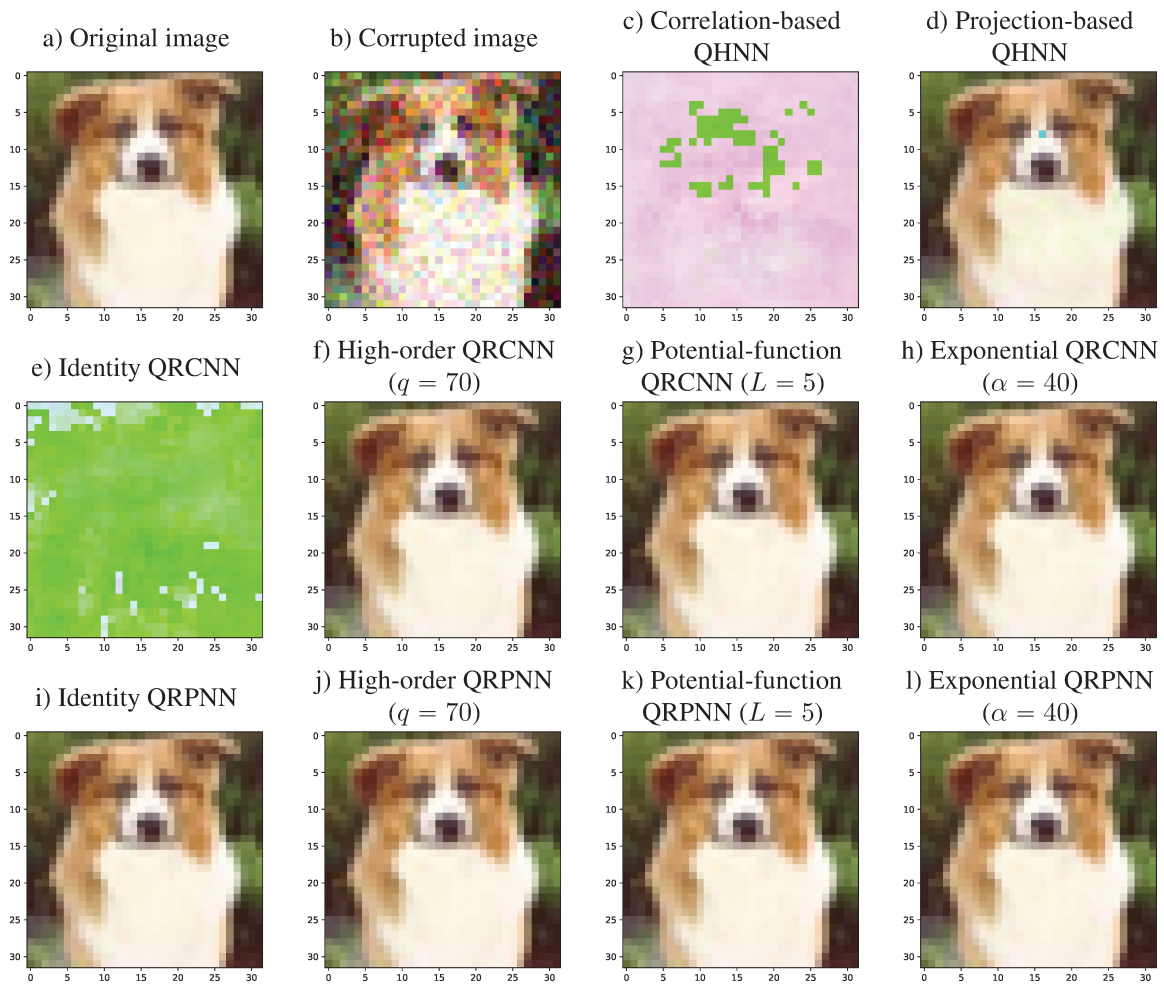


Figure 10 – Original color image, input image corrupted by Gaussian noise with standard deviation 0.1, and the corresponding images retrieved by the quaternion-valued associative memories.

near the dog's nose. All the other associative memories succeed to retrieve the original image. Quantitatively, we say that an associative memory manages to remember a stored image if the error given by the Euclidean norm $\|\mathbf{u}^1 - \mathbf{y}\|$, where $\mathbf{y}$ denotes the retrieved quaternion-valued vector, is less than or equal to a tolerance $\tau = 10^{-4}$. Table 1 shows the error produced by the QHNN, QRCNN, and QRPNN memory models. The table also contains the error between the fundamental memory $\mathbf{u}^1$ and the quaternion-valued vector corresponding to the corrupted image.

For a better comparison of the noise tolerance of the quaternion-valued associative memories, we repeated the preceding experiment 100 times. We also considered images corrupted by Gaussian noise with several different standard deviation values. Figure 11 shows the probability of successful recall by the standard deviation of the Gaussian noise introduced in the input image. In agreement with Theorem 2.1, the QRPNN always managed to remember undistorted patterns (zero standard deviation). Furthermore, like the experiment described in the previous subsection, the projection-based QHNN differs from the identity QRPNN. The

Table 1 – Absolute error between the fundamental memory $\mathbf{u}^1$ and either the input or the quaternion-valued vector recalled by an associative memory model.

Corrupted image:	19.69
Correlation-based QHNN:	29.87
Projection-based QHNN:	1.11
Identity QRCNN:	3.67×10^{1}
High-order QRCNN:	5.82×10^{-9}
Potential-function QRCNN:	3.17×10^{-15}
Exponential QRCNN:	1.78×10^{-14}
Identity QRPNN:	1.31×10^{-4}
High-order QRPNN:	5.82×10^{-9}
Potential-function QRPNN:	4.08×10^{-15}
Exponential QRPNN:	1.78×10^{-14}

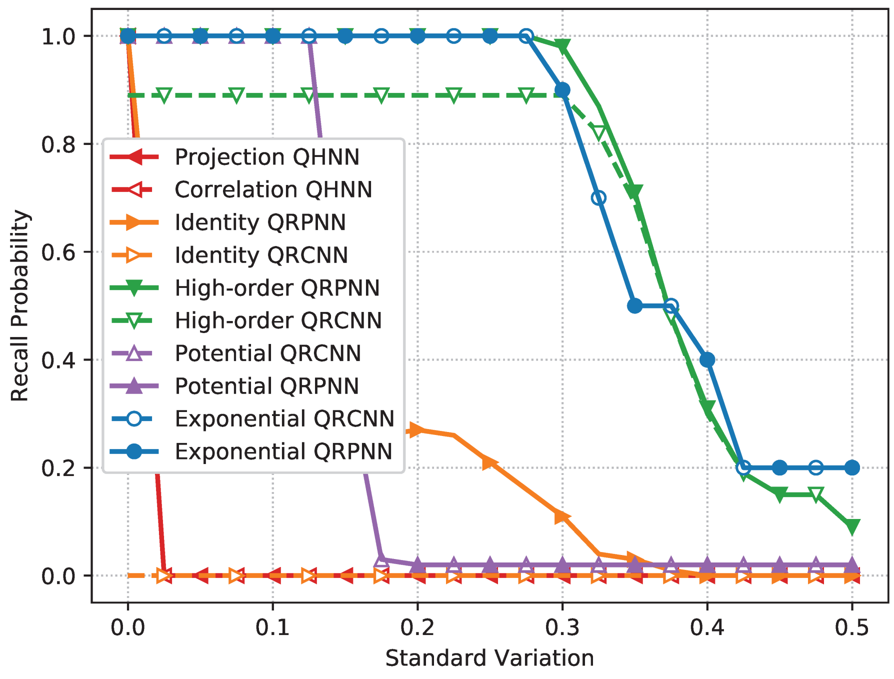


Figure 11 – Recall probability of quaternion-valued associative memories by the standard deviation of the Gaussian noise introduced in the input.

latter, however, yielded larger recall probabilities because it circumvents the rotational invariance present in the QHNN model (Kobayashi, 2016a).

Finally, we repeated the experiment used to generate Figure 11 considering only the exponential QRPNN and QRCNN but with different values of the parameter α. Moreover, to better discriminate the QRPNN and the QRCNN models, instead of computing the recall probability, we computed the Euclidean error between the desired output $\mathbf{u}^1$ and the retrieved vector $\mathbf{y}$, that is, the error is given by $\|\mathbf{u}^1 - \mathbf{y}\|_2$. Figure 12 shows the mean error by the standard

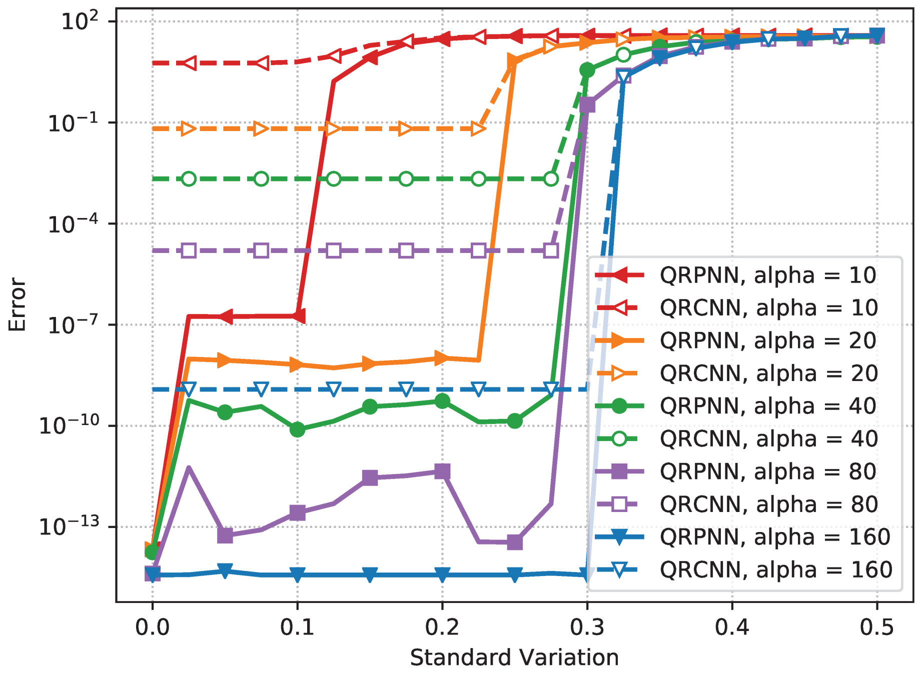


Figure 12 – Error between the desired output and the retrieved quaternion-valued vector by the standard deviation of the Gaussian noise introduced in the input.

deviation of the Gaussian noise introduced in the original color image. Note that the error produced by the exponential QRPNNs from an undistorted input are all around the machine precision, that is, around 10^{-14}. Equivalently, the QRPNNs succeeded to recall undistorted images. Finally, note also that the error produced by both QRPNN and QRCNN associative memories decreases as the parameter α increases. Nevertheless, the average error produced by the QRPNN are always below the corresponding QRCNN models.

2.4 Conclusions

In this chapter we have proposed the quaternion-valued recurrent projection neural network (QRPNN), a new associative memory model which combines the projection storing rule with the QRCNN model. We showed that the new model presents optimal storage capacity and higher noise tolerance than the QRCNN as well as the traditional Hopfield models. In the next chapter, we extend the QRCNN and QRPNN models to a broad class of hypercomplex number systems.

3 Hypercomplex-Valued Models

In this chapter, we extend the RCNNs and RPNNs to a broad class of hypercomplex number systems. Initially, we present a background on hypercomplex numbers and some fundamental definitions necessary to define the extensions of QRCNNs and QRPNNs, including a brief introduction to the Hypercomplex-Valued Discrete-Time Hopfield Neural Networks (HHNNs). Then, we present the Hypercomplex Recurrent Correlation Neural Networks (HRCNNs) and the Hypercomplex Recurrent Projection Neural Networks (HRPNNs). Besides, we study the dynamics of HRCNNs and HRPNNs. Finally, we perform computational experiments in a broad class of hypercomplex number systems, applying these models for pattern reconstruction.

3.1 Background on Hypercomplex Numbers

Hypercomplex numbers generalize the notion of complex and hyperbolic numbers as well as quaternions, tessarines, octonions, and many other high-dimensional algebras including Clifford and Cayley-Dickson algebras (Brown, 1967; Shenitzer et al., 1989; Delangue et al., 1992; Ell and Sangwine, 2007; Hitzer et al., 2013; Vaz and da Rocha, 2016).

A hypercomplex number over the real field is written as

$$p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n, \tag{3.1}$$

where n is a non-negative integer, $p_0, p_1, \ldots, p_n$ are real numbers, and the symbols $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_n$ are the hyperimaginary units (Shenitzer et al., 1989; de Castro and Valle, 2020). The set of all hypercomplex numbers given by (3.1) is denoted in this thesis by $\mathbb{H}$. Examples of hypercomplex numbers include the complex numbers $\mathbb{C}$, hyperbolic numbers $\mathbb{U}$, dual numbers $\mathbb{D}$, quaternions $\mathbb{Q}$, tessarines (also called commutative quaternions) $\mathbb{T}$, and octonions $\mathbb{O}$.

We would like to point out that a real number $\alpha \in \mathbb{R}$ can be identified with the hypercomplex number $\alpha + 0\mathbf{i}_1 + \ldots + 0\mathbf{i}_n \in \mathbb{H}$. Furthermore, the real-part of a hypercomplex number $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n$ is the real-number defined by $\text{Re}\{p\} := p_0$.

A hypercomplex number *system* is a set of hypercomplex numbers equipped with an *addition* and a *multiplication* (or *product*). The addition of two hypercomplex numbers $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n$ and $q = q_0 + q_1\mathbf{i}_1 + \ldots + q_n\mathbf{i}_n$ is defined in a component-wise manner according to the expression

$$p + q = (p_0 + q_0) + (p_1 + q_1)\mathbf{i}_1 + \ldots + (p_n + q_n)\mathbf{i}_n. \tag{3.2}$$

The product between p and q, denoted by the juxtaposition of p and q, is defined using the distributive law and a multiplication table. Precisely, the multiplication table specifies the product

Table 2 – Multiplication tables for complex numbers and quaternions.

$\times$	$\mathbf{i}_1$
$\mathbf{i}_1$	-1

$\times$	$\mathbf{i}_1$	$\mathbf{i}_2$	$\mathbf{i}_3$
$\mathbf{i}_1$	-1	$\mathbf{i}_3$	$-\mathbf{i}_2$
$\mathbf{i}_2$	$-\mathbf{i}_3$	-1	$\mathbf{i}_1$
$\mathbf{i}_3$	$\mathbf{i}_2$	$-\mathbf{i}_1$	-1

of any two hyperimaginary units:

$$\mathbf{i}_\mu \mathbf{i}_\nu = a_{\mu\nu,0} + a_{\mu\nu,1}\mathbf{i}_1 + \ldots + a_{\mu\nu,n}\mathbf{i}_n, \quad \forall \mu,\nu \in \{1,\ldots,n\}. \tag{3.3}$$

For example, Table 2 shows the multiplication table of complex numbers and quaternions. On the left we show the table for complex numbers, and on the right for quaternions. A hypercomplex number system is characterized by its multiplication table. Then, given two hypercomplex numbers p and q, we use the distributive law and the corresponding multiplication table, obtaining

$$\begin{aligned} pq = \left(p_0q_0 + \sum_{\mu,\nu=1}^{n} p_\mu q_\nu a_{\mu\nu,0}\right) + \left(p_0q_1 + p_1q_0 + \sum_{\mu,\nu=1}^{n} p_\mu q_\nu a_{\mu\nu,1}\right)\mathbf{i}_1 + \ldots \\ + \left(p_0q_n + p_nq_0 + \sum_{\mu,\nu=1}^{n} p_\mu q_\nu a_{\mu\nu,n}\right)\mathbf{i}_n. \end{aligned} \tag{3.4}$$

We would like to point out that we can identify a hypercomplex number $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n$ with the $(n+1)$-tuple $(p_0, p_1, \ldots, p_n)$ of real numbers. Furthermore, a hypercomplex number system can be embedded in a $(n+1)$-dimensional vector space where the vector addition and the scalar multiplication are obtained from (3.2) and (3.4), respectively. In view of this fact, we write $\mathtt{dim}(\mathbb{H}) = n+1$. Moreover, the set $\mathbb{H}$ of all hypercomplex numbers inherits the topology from $\mathbb{R}^{n+1}$ (de Castro and Valle, 2020). Borrowing the terminology of linear algebra, we speak of a linear operator $T : \mathbb{H} \to \mathbb{H}$ if $T(\alpha p + q) = \alpha T(p) + T(q)$, for all $p, q \in \mathbb{H}$ and $\alpha \in \mathbb{R}$. A linear operator that is an involution and also an antihomomorphism is called a *reverse-involution* (Ell and Sangwine, 2007). Formally, we have the following definition:

Definition 1 (Reverse-involution (de Castro and Valle, 2020))**.** *An operator $\tau : \mathbb{H} \to \mathbb{H}$ is a reverse-involution if*

$$\tau\big(\tau(p)\big) = p, \tag{3.5}$$

$$\tau(pq) = \tau(q)\tau(p), \tag{3.6}$$

$$\tau(\alpha p + q) = \alpha\tau(p) + \tau(q), \tag{3.7}$$

for all $p, q \in \mathbb{H}$ and $\alpha \in \mathbb{R}$.

The natural conjugate of a hypercomplex number $p = p_0 + p_1\mathbf{i}_1 + \cdots + p_n\mathbf{i}_n$, denoted by $\bar{p}$, is defined by

$$\bar{p} = p_0 - p_1\mathbf{i}_1 - \cdots - p_n\mathbf{i}_n. \tag{3.8}$$

The *natural conjugation* is an example of a reverse-involution in some hypercomplex number systems such as the complex and quaternion number systems. The identity mapping $\tau(p) = p$, for all $p \in \mathbb{H}$, is also a reverse-involution if the multiplication is commutative. In this case, the identity is referred to as the *trivial reverse-involution*. Other examples of reverse-involutions include the *quaternion anti-involutions* in quaternion algebra and the *Clifford conjugation* in Clifford algebras (Ell and Sangwine, 2007; Delangue et al., 1992; Vaz and da Rocha, 2016).

A reverse-involution allows us to define a symmetric bilinear form $\mathcal{B} : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ by means of the equation:

$$\mathcal{B}(p, q) = \text{Re}\{\tau(p)q\}, \quad \forall p, q \in \mathbb{H}. \tag{3.9}$$

Intuitively, $\mathcal{B}$ measures a relationship between p and q by taking into account the algebraic properties of the multiplication and the reverse-involution τ. For example, the symmetric bilinear form $\mathcal{B}$ coincides with the usual inner product on complex numbers, quaternions, and octonions with the natural conjugation. The symmetric bilinear form $\mathcal{B}$ plays an important role in the definition of the hypercomplex-valued recurrent correlation neural networks.

Finally, let us also recall the following class of hypercomplex-valued functions:

Definition 2 ($\mathcal{B}$ function (de Castro and Valle, 2020))**.** *Consider a hypercomplex number system $\mathbb{H}$ equipped with a reverse-involution τ and let $\mathcal{D} \subset \mathbb{H}$, $\mathcal{S} \subset \mathbb{H}$, and $\mathcal{B} : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ be the symmetric bilinear form defined by* (3.9)*. A hypercomplex-valued function $\phi : \mathcal{D} \to \mathcal{S}$ is called a $\mathcal{B}$ function if*

$$\mathcal{B}(\phi(q), q) > \mathcal{B}(s, q), \quad \forall q \in \mathcal{D}, \forall s \in \mathcal{S} \backslash \{\phi(q)\}. \tag{3.10}$$

In words, a $\mathcal{B}$ function $\phi : \mathcal{D} \to \mathcal{S}$ projects a hypercomplex number $q \in \mathcal{D}$ onto $\mathcal{S}$ with respect to the symmetric bilinear form $\mathcal{B}$. Indeed, according to $\mathcal{B}$, $\phi(q)$ is more related to q than any other element $s \in \mathcal{S}$. Examples of $\mathcal{B}$ functions include the complex-valued signum function for complex-valued Hopfield neural networks, the function that normalizes its arguments to length one on Cayley-Dickson algebras with the natural conjugation, and the split-sign functions for some real Clifford algebras with Clifford conjugation (de Castro and Valle, 2020; Aizenberg and Aizenberg, 1992; Jankowski et al., 1996).

3.2 A Review of the Hypercomplex-Valued Hopfield Neural Networks

Hypercomplex-valued versions of the Hopfield network have been extensively investigated in the past years using complex numbers (Jankowski et al., 1996; Lee, 2006), dual numbers (Kuroe et al., 2011), hyperbolic numbers (Kobayashi, 2013), tessarines (Isokawa et al., 2010; Kobayashi, 2018a), quaternions (Isokawa et al., 2008b; Valle and Castro, 2018), octonions (Kuroe and Iima, 2016), and further hypercomplex number systems (Vallejo and

Bayro-Corrochano, 2008; Kuroe, 2013; Popa, 2016). In this section, we briefly review the broad class of hypercomplex-valued Hopfield neural networks (HCNNs), which includes the complex-valued and quaternion-valued HCNN models as particular instances (Valle, 2014a, 2018).

In order to analyze the stability of the HCNNs, we use part of the theory present in the work of de Castro and Valle (2020), which introduced the following class of hypercomplex number systems:

Definition 3 (Real-Part Associative Hypercomplex Number Systems)**.** *A hypercomplex number system equipped with a reverse-involution τ is called a real-part associative hypercomplex number system (Re-AHN) if the following identity holds true for any three of its elements p, q, r:*

$$Re\left\{(pq)r - p(qr)\right\} = 0. \tag{3.11}$$

In particular, we speak of a positive semi-definite (or non-negative definite) real-part associative hypercomplex number system if the symmetric bilinear form $\mathcal{B}$ given by (3.9) *satisfies $\mathcal{B}(p,p) \geqslant 0$, $\forall p \in \mathbb{H}$.*

Complex numbers, quaternions, and octonions are examples of real-part associative hypercomplex number systems with the natural conjugation. Real-part associative hypercomplex number systems also include the tessarines, Cayley-Dickson algebras, and real Clifford algebras (de Castro and Valle, 2020).

Let $\mathbb{H}$ be a real-part associative hypercomplex number system and $\phi : \mathcal{D} \to \mathcal{S}$ be a function. In analogy to the traditional discrete-time Hopfield network, let $w_{ij} \in \mathbb{H}$ denote the jth hypercomplex-valued synaptic weight of the ith neuron of a network with N neurons. Also, let the state of the network at time t be represented by a hypercomplex-valued column-vector $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^T \in \mathcal{S}^N$, that is, $x_i(t) = x_{i0}(t) + x_{i1}(t)\mathbf{i}_1 + \ldots + x_{in}(t)\mathbf{i}_n$ corresponds to the state of the ith neuron at time t. Given an initial state (or input vector) $\mathbf{x}(0) = [x_1(0), \ldots, x_N(0)]^T \in \mathcal{S}^N$, the hypercomplex-valued Hopfield network defines recursively the sequence $\{\mathbf{x}(t)\}_{t \geqslant 0}$ by means of the equation

$$x_i(t + \Delta t) = \begin{cases} \phi\big(a_i(t)\big), & a_i(t) \in \mathcal{D}, \\ x_i(t), & \text{otherwise}, \end{cases} \tag{3.12}$$

where $a_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t)$ is the hypercomplex-valued activation potential of the ith neuron at time t.

Like the traditional real-valued Hopfield network, the sequence produced by (3.12) is convergent in asynchronous update mode if the synaptic weights satisfy $w_{ij} = \tau(w_{ji})$ and one of the two cases below holds true (de Castro and Valle, 2020):

1. $w_{ii} = 0$ for any $i \in \{1, \ldots, N\}$.
2. w_{ii} is a nonnegative real number for any $i \in \{1, \ldots, N\}$ and $\mathbb{H}$ is a positive semi-definite real-part associative hypercomplex number system.

These two conditions generalize the convergence stability criterion reviewed in Chapter 1 for real and complex algebras as well as the stability criterion for the CV-QHNN in Chapter 2.1. In the real case τ is the identity reverse-involution. In the complex and quaternion cases τ is the natural conjugation.

3.3 Hypercomplex Recurrent Correlation Neural Networks

In the following, we generalize further the RCNNs for the storage and recall of hypercomplex-valued vectors. Let $\mathbb{H}$ be a hypercomplex number system, $\phi : \mathcal{D} \to \mathcal{S}$ be a $\mathcal{B}$ function, and $f : \mathbb{R} \to \mathbb{R}$ be a real-valued continuous and monotonic non-decreasing function. Given a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, in which

$$u_i^\xi = u_{i0}^\xi + u_{i1}^\xi \mathbf{i}_1 + \ldots + u_{in}^\xi \mathbf{i}_n \in \mathcal{S}, \quad \forall i = 1, \ldots, N, \forall \xi = 1, \ldots, P, \tag{3.13}$$

are hypercomplex numbers, a hypercomplex-valued RCNN (HRCNN) defines recursively the following sequence of hypercomplex-valued vectors in $\mathcal{S}^N$ for $t \geqslant 0$ and $i = 1, \ldots, N$:

$$x_i(t + \Delta t) = \begin{cases} \phi\big(a_i(t)\big), & a_i(t) \in \mathcal{D}, \\ x_i(t), & \text{otherwise}, \end{cases} \tag{3.14}$$

where the activation potential of the ith output neuron at time t is given by

$$a_i(t) = \sum_{\xi=1}^{P} w_\xi(t) u_i^\xi, \quad \forall i = 1, \ldots, N, \tag{3.15}$$

with

$$w_\xi(t) = f\left(\sum_{i=1}^{N} \mathcal{B}\big(u_i^\xi, x_i(t)\big) \right), \quad \forall \xi \in 1, \ldots, P. \tag{3.16}$$

Examples of HRCNNs include the generalizations of the bipolar (real-valued) RCNNs of Chiueh and Goodman (1991), and its complex-valued and quaternion valued generalizations (Valle, 2014a, 2018) viewed in the previous chapters.

Analogously to the real case, we define the *identity* HRCNN, the *high-order* HRCNN, the *potential-function* HRCNN, and the *exponential* HRCNN using the weights given in (3.16) with the identity function, and the functions given by (1.26), (1.27) and (1.28) for the corresponding hypercomplex number system domain.

Let us now study the stability of HRCNNs. The following theorem shows that a HRCNN yields a convergent sequence $\{\mathbf{x}(t)\}_{t \geqslant 0}$ of hypercomplex-valued vectors independently of the initial state vector $\mathbf{x}(0) \in \mathcal{S}^N$.

Theorem 3.1. *Let $\mathbb{H}$ be a hypercomplex number system, $\phi : \mathcal{D} \to \mathcal{S}$ be a $\mathcal{B}$ function where $\mathcal{S}$ is a compact set, and $f : \mathbb{R} \to \mathbb{R}$ be a real-valued continuous and monotonic non-decreasing function. Given a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^P\} \subset \mathcal{S}^P$, the sequence $\{\mathbf{x}(t)\}_{t \geqslant 0}$ given by (3.14), (3.15), and (3.16) is convergent for any initial state $\mathbf{x}(0) \in \mathcal{S}^N$ using either asynchronous (sequential) or synchronous update mode.*

Remark 1. *In contrast to the hypercomplex-valued Hopfield networks, the stability analysis of the hypercomplex-valued recurrent correlation neural networks does not require that $\mathbb{H}$ is a real-part associative hypercomplex-number system. As a consequence, hypercomplex-valued RCNNs can be defined in more general class of hypercomplex number systems.*

Remark 2. *As usual, Theorem 3.1 is derived by showing that the function $E : \mathcal{S}^N \to \mathbb{R}$ given by*

$$E(\mathbf{x}) = -\sum_{\xi=1}^{P} F\left(\sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i)\right), \quad \forall \mathbf{x} \in \mathcal{S}^N, \tag{3.17}$$

where $F : \mathbb{R} \to \mathbb{R}$ is a primitive of f, is an energy function of the HRCNN.

Proof. First of all, let $F : \mathbb{R} \to \mathbb{R}$ be the primitive of the continuous function f given by

$$F(x) = \int_0^x f(t)dt, \quad x \in \mathbb{R}. \tag{3.18}$$

From the mean value theorem and the monotonicity of f, we obtain:

$$F(y) - F(x) \geqslant f(x)(y - x), \quad \forall x, y \in \mathbb{R}. \tag{3.19}$$

Let us show that the function $E : \mathcal{S}^N \to \mathbb{R}$ given by (3.17) is an energy function of the HRCNN, that is, E is a real-valued bounded function whose values decrease along non-stationary trajectories. Let us first show that E is a bounded function. Recall that the set of hypercomplex numbers $\mathbb{H}$ inherits the topology from $\mathbb{R}^{n+1}$ by identifying $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n$ with the $(n+1)$-tuple $(p_0, p_1, \ldots, p_n)$. A well-known generalization of the extreme value theorem from calculus for continuous functions establishes that the continuous image of a compact set is compact. Thus, as a symmetric bilinear form on a finite dimensional vector space, $\mathcal{B}$ is continuous. Also, since f is continuous, its primitive F exists and is continuous. Therefore, the function E given by (3.17) is continuous because it is the sum and the composition of continuous functions. In addition, since $\mathcal{S}$ is compact, the Cartesian product $\mathcal{S}^N = \mathcal{S} \times \ldots \times \mathcal{S}$ is also compact. Hence, from the extreme value theorem, we conclude that the function E given by (3.17) is bounded.

Let us now show that E decreases along non-stationary trajectories. To simplify notation, let $\mathbf{x} \equiv \mathbf{x}(t)$ and $\mathbf{x}' \equiv \mathbf{x}(t + \Delta t)$ for some $t \geqslant 0$. Furthermore, let us suppose that a set of neurons changed their state at time t. Formally, suppose that $x_i' \neq x_i$ if $i \in \mathcal{I}$ and $x_i' = x_i$ for all $i \notin \mathcal{I}$, where $\mathcal{I} \subseteq \{1, \ldots, N\}$ is a non-empty set of indexes. Using (3.19), we conclude that the variation of E satisfies the following inequality:

$$\begin{aligned} \Delta E &= E(\mathbf{x}') - E(\mathbf{x}) \\ &= -\sum_{\xi=1}^{P} \left[F\left(\sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i') \right) - F\left(\sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i) \right) \right] \\ &\leqslant -\sum_{\xi=1}^{P} f\left(\sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i) \right) \left[\sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i') - \sum_{i=1}^{N} \mathcal{B}(u_i^\xi, x_i) \right]. \end{aligned} \tag{3.20}$$

From (3.16), recalling that $\mathcal{B}$ is a symmetric bilinear form, and using (3.15), we obtain

$$\begin{aligned} \Delta E &\leqslant -\sum_{\xi=1}^{P} w_\xi(t) \sum_{i \in \mathcal{I}} \mathcal{B}(u_i^\xi, x_i' - x_i) \\ &= -\sum_{i \in \mathcal{I}} \mathcal{B}\left(\sum_{\xi=1}^{P} w_\xi(t) u_i^\xi, (x_i' - x_i) \right) \\ &= -\sum_{i \in \mathcal{I}} \mathcal{B}(a_i(t), x_i' - x_i) \\ &= -\sum_{i \in \mathcal{I}} [\mathcal{B}(a_i(t), x_i') - \mathcal{B}(a_i(t), x_i)] = -\sum_{i \in \mathcal{I}} \Delta_i. \end{aligned} \tag{3.21}$$

Now, if the ith neuron changed its state at time t, then

$$x_i' = x_i(t + \Delta t) = \phi(a_i(t)) \neq x_i(t). \tag{3.22}$$

Since ϕ is a $\mathcal{B}$ function, we have

$$\Delta_i = \mathcal{B}\Big(a_i(t), \phi\big(a_i(t)\big)\Big) - \mathcal{B}\Big(a_i(t), x_i(t)\Big) > 0, \quad \forall i \in \mathcal{I}. \tag{3.23}$$

The latter inequality implies that $\Delta E < 0$ if at least one neuron changes its state at time t, which concludes the proof of the theorem. □

3.4 Hypercomplex Recurrent Projection Neural Networks

In section 3.3, we extended the RCNN model to a broad class of hypercomplex numbers. Following this reasoning, in this section, we extend the QRPNNs to hypercomplex number systems. These new models are called hypercomplex-valued recurrent projection neural networks (HRPNNs). As an initial work, we briefly study the dynamical behavior and the noise tolerance of the HRPNNs. Analogously to the HRCNNs, let $\mathbb{H}$ be a hypercomplex number system, $\phi : \mathcal{D} \to \mathcal{S}$ be a $\mathcal{B}$ function, and $f : \mathbb{R} \to \mathbb{R}$ be a real-valued continuous and monotonic non-decreasing function. Given a fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, in which

$$u_i^\xi = u_{i0}^\xi + u_{i1}^\xi \mathbf{i}_1 + \ldots + u_{in}^\xi \mathbf{i}_n \in \mathcal{S}, \quad \forall i = 1, \ldots, N, \forall \xi = 1, \ldots, P, \tag{3.24}$$

are hypercomplex numbers, define

$$c_{\eta\xi} = f\left(\sum_{i=1}^{N} \mathcal{B}\big(u_i^\eta, u_i^\xi\big)\right), \quad \forall \eta, \xi \in \{1, \ldots, P\}. \tag{3.25}$$

and

$$v_i^\xi = \sum_{\eta=1}^{P} u_i^\eta c_{\eta\xi}^{-1} \quad \forall \xi = 1, \ldots, P \quad \text{and} \quad i = 1, \ldots, N, \tag{3.26}$$

where $c_{\eta\xi}^{-1}$ denotes the entries of the inverse of $C = (c_{\eta\xi}) \in \mathbb{R}^{P\times P}$ given by (3.25). Then, a hypercomplex-valued recurrent projection neural network (HRPNN) defines recursively the following sequence of hypercomplex-valued vectors in $\mathcal{S}^N$ for $t \geqslant 0$ and $i = 1, \ldots, N$:

$$x_i(t+\Delta t) = \begin{cases} \phi\big(a_i(t)\big), & a_i(t) \in \mathcal{D}, \\ x_i(t), & \text{otherwise}, \end{cases} \tag{3.27}$$

where the activation potential of the ith output neuron at time t is given by

$$a_i(t) = \sum_{\xi=1}^{P} w_\xi(t) v_i^\xi, \quad \forall i = 1, \ldots, N, \tag{3.28}$$

and

$$w_\xi(t) = f\left(\sum_{i=1}^{N} \mathcal{B}\big(u_i^\xi, x_i(t)\big)\right), \quad \forall \xi \in 1, \ldots, P. \tag{3.29}$$

Note that (3.28) is directly related with (3.15). The HRPNN model is a variation of the HRCNN, in which the weights of the output layer are defined inspired by the projection rule. The matrix C in (3.26) and the weights in (3.29) are the core elements of the extended model. The symmetric bilinear form generalizes the C matrix and weights w given in equations (2.21) and (2.22), respectively, in chapter 2.

3.5 Examples and Computational Experiments

3.5.1 HRCNNs synthetic experiments

Let us now provide some examples and perform some computational experiments with hypercomplex-valued exponential correlation neural networks (ECNN). See the appendix for some details of our implementation of the hypercomplex-valued ECNNs.

We would like to point out that we refrained to consider other HRCNNs besides the exponential due to the following: First, there is a vast literature on real-valued RCNN models based on the exponential function f_e (Chiueh and Goodman, 1991; Hassoun and Watta, 1996; Hancock and Pelillo, 1998; Perfetti and Ricci, 2008). The exponential, high-order, and potential-function are non-negative parametric functions continuous and strictly increasing on the variable x and characterized by an exponential on their parameter (Valle, 2018). Therefore,

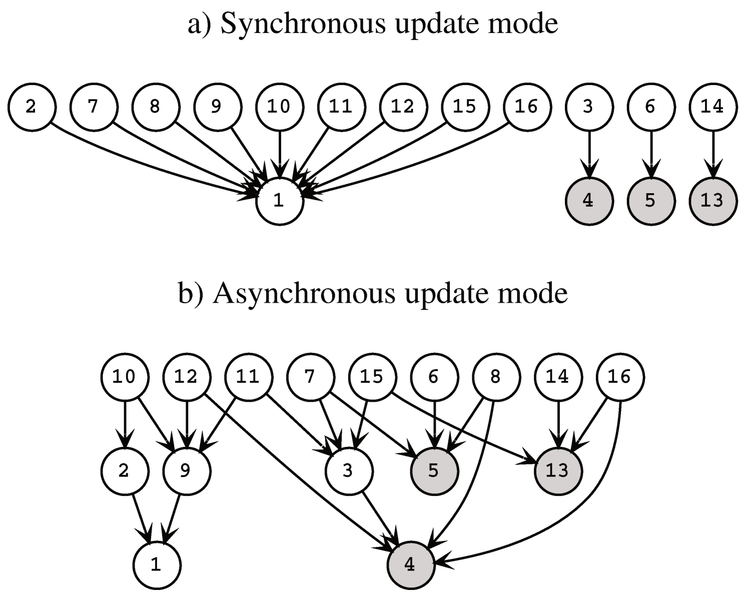


Figure 13 – Dynamic of the complex-valued multistate ECNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes.

the three functions f_e, f_h, and f_p belong to the same family of functions. According to Chiueh and Goodman (1991), the bipolar (real-valued) ECNN seems to be the RCNN most ameable to VLSI implementation. In fact, the storage capacity and noise tolerance of the bipolar ECNN have been extensively investigated (Chiueh and Goodman, 1991). In addition, the storage capacity of the bipolar ECNN can reach the capacity of an ideal associative memory (Hassoun and Watta, 1996). Finally, besides being closely related to support vector machines and the kernel trick (Perfetti and Ricci, 2008), the bipolar ECNN has a Bayesian interpretation (Hancock and Pelillo, 1998).

3.5.1.1 Complex-Valued Multistate RCNNs

As far as we know, the first hypercomplex-valued neural networks which have been used to implement associative memories dates to the late 1980s (Noest, 1988a,b; Jankowski et al., 1996). As was mentioned in section 1.1.3, the activation function that make difference for complex models is the complex-signum function. Using the polar representation of complex numbers, de Castro and Valle (2020) showed that the complex-valued signum function given by (1.21) is a $\mathcal{B}$-function in the system of complex numbers with the natural conjugation. Therefore, from Theorem 3.1, we conclude that the complex-valued multistate RCNNs given by (3.14), (3.15), and (3.16) with $\phi \equiv \operatorname{csgn}$ always yield a convergent sequence of complex-valued vectors. We would like to point out that this remark is a new contribution to the field because the complex-valued recurrent correlation neural network introduced in (Valle, 2014a) is based on the continuous-valued function given by $f(z) = z/|z|$ for all complex number $z \neq 0$. Now we are using the csgn as a new proposal of activation function.

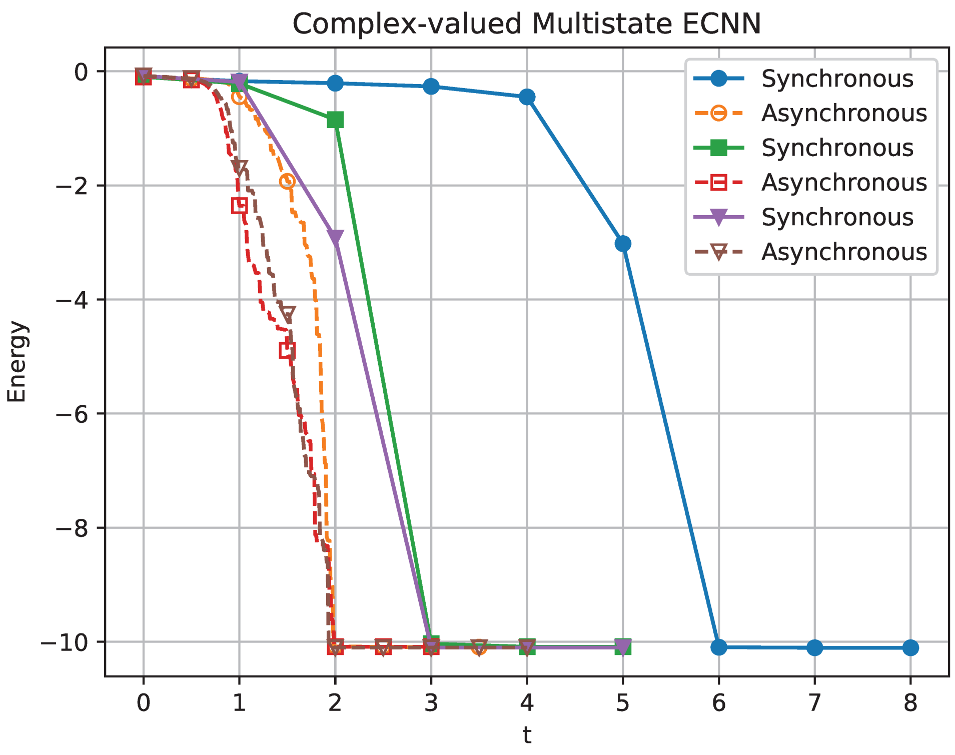


Figure 14 – Evolution of the energy of the complex-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ with a resolution factor $K = 256$. The curves marked by "●" and "○", "■" and "□", as well as "▾" and "▽" refer to pairs of ECNNs that differ only from the update mode.

Example 1. *Consider a resolution factor $K = 4$ and the fundamental memory set*

$$\mathcal{U} = \left\{ \mathbf{u}^1 = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix}, \mathbf{u}^2 = \begin{bmatrix} \mathbf{i} \\ 1 \end{bmatrix}, \mathbf{u}^3 = \begin{bmatrix} -\mathbf{i} \\ 1 \end{bmatrix} \right\} \subset \mathcal{S}^2, \tag{3.30}$$

where $\mathcal{S} = \{1, \mathbf{i}, -1, -\mathbf{i}\}$ is the set given by (1.20). Figure 13 portrays the entire dynamic of the complex-valued multistate ECNN designed for the storage of the fundamental memory $\mathcal{U}$ given by (3.30). In this example, we considered $\alpha = 1/2$, $\beta = 1$, and both synchronous and asynchronous updates. In these directed graph, a node corresponds to a state of the neural network while an edge from node i to node j means that we can obtain the jth state from the ith state evolving (3.14). Also, there are 16 possible states and the fundamental memories, which corresponds to the 4th, 5th, and 13th states, correspond to gray nodes in Figure 13. In agreement with Theorem 3.1, the exponential complex-valued multistate RCNN always settled at an equilibrium using either synchronous or asynchronous update. Moreover, both synchronous and asynchronous multistate models have one spurious memory; the vector $\boldsymbol{s}^1 = [1, 1] \in \mathcal{S}^2$, which corresponds to the 1st state.

Experiment 1. *We synthesized complex-valued multistate ECNN models for the storage and recall of fundamental memories $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, where $N = 100$ and $P = 160$, using*

$\alpha = 10/N$ and $\beta = e^{-10}$. Furthermore, we adopted a resolution factor $K = 256$ and randomly generated the components of a fundamental memory, as well as the input patterns, using uniform distribution (the elements of $\mathcal{S}$ given by (1.20) have all equal probability). Figure 14 shows the evolution of the energy of the complex-valued ECNNs operating synchronously and asynchronously. In agreement with Theorem 3.1, the energy always decreases until the network settles at an equilibrium. As in the bipolar case, although the ECNNs with asynchronous update usually comes to rest at a stationary state earlier than the ECNN with synchronous update, the computational implementation of the latter is usually faster than the asynchronous model. Furthermore, since the set $\mathcal{D}$ given by (1.19) is dense on the complex-numbers $\mathbb{C}$, we refrained to check if the argument of csgn *given by (1.21) belongs to $\mathcal{D}$ in the computational implementation of the complex-valued multistate ECNNs.*

3.5.2 Hyperbolic-Valued Multistate RCNNs

Neural networks based on hyperbolic numbers constitute an active topic of research since the earlier 2000s (Buchholz and Sommer, 2000; Ontrup and Ritter, 2001; Nitta and Kuroe, 2018). Hyperbolic-valued Hopfield neural networks, in particular, have been extensively investigated in the last decade by Kobayashi, Kuroe, and collaborators (Kuroe et al., 2011; Kobayashi, 2013, 2016b,c, 2018c, 2019). Hyperbolic-valued Hopfield neural networks are usually synthesized using either Hebbian learning (Kobayashi, 2013, 2019) or the projection rule (Kobayashi, 2019). On one hand, the hyperbolic-valued Hopfield neural network with the Hebbian learning is very similar to the complex-valued model and, thus, they suffer from a low storage capacity due to the cross-talk between the stored items (Jankowski et al., 1996; Müezzinoğlu et al., 2003; Kobayashi, 2019). On the other hand, the projection rule may fails to satisfy the stability conditions imposed on the synaptic weights (Kobayashi, 2019). Examples of the activation function employed on hyperbolic-valued Hopfield neural networks include the split-sign function (Kobayashi, 2016b) and the directional multistate activation function (Kobayashi, 2018c). In the following, we address the stability of hyperbolic-valued RCNNs with the directional multistate activation function, which corresponds to the csgn function given by (1.21).

Such as the complex-numbers, hyperbolic numbers are 2-dimensional hypercomplex numbers of the form $p = p_0 + p_1\mathbf{i}$. In contrast to the complex imaginary unit, the hyperbolic unit satisfies $\mathbf{i}^2 = 1$. The set of all hyperbolic numbers is denoted by $\mathbb{U}$. The (natural) conjugate of a hyperbolic number $p = p_0 + p_1\mathbf{i}$ is $\bar{p} = p_0 - p_1\mathbf{i}$. Using the natural conjugate, the symmetric bilinear form given by (3.9) for hyperbolic numbers satisfies $\mathcal{B}(p,q) = p_0q_0 - p_1q_1$, for all $p,q \in \mathbb{U}$. Unfortunately, the csgn function given by (1.21) is not a $\mathcal{B}$-function. Thus, a hyperbolic-valued RCNN may fail to settle at an equilibrium. The following example and computational experiment confirm this remark.

Example 2. *Consider the fundamental memory set $\mathcal{U}$ given by (3.30), where $\mathcal{S} = \{1, \mathbf{i}, -1, -\mathbf{i}\}$*

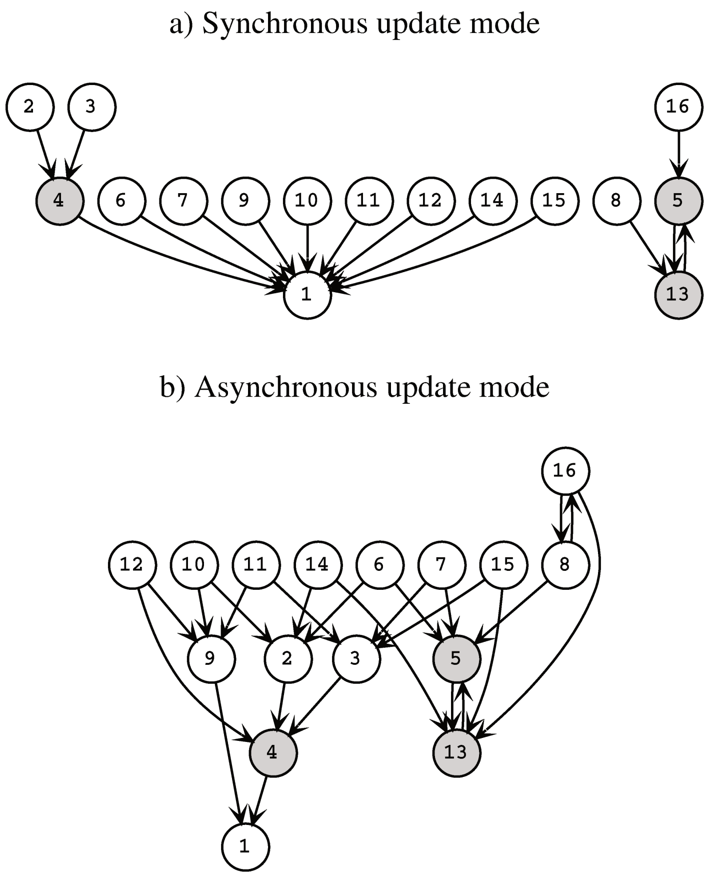


Figure 15 – Dynamic of the hyperbolic-valued multistate ECNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes.

is the set given by (1.20) *and* $\mathbf{i}$ *denotes the hyperbolic unit, i.e.,* $\mathbf{i}^2 = 1$*. Using* $\alpha = 1$ *and* $\beta = 1$*, we synthesized hyperbolic-valued mustistate ECNN models designed for the storage of the fundamental memory set* $\mathcal{U}$*. Such as in the previous examples, Figure* 15 *depicts the entire dynamic of the hyperbolic-valued ECNN models obtained using synchronous and asynchronous update mode. Note that the network, using either synchronous or asynchronous update, may cycle between the 5th and 13th states, which correspond respectively to the fundamental memories* $\mathbf{u}^2$ *and* $\mathbf{u}^3$.

Experiment 2. *Like in the previous experiments, we synthesized hyperbolic-valued ECNN models using* $\alpha = 10/N$ *and* $\beta = e^{-10}$ *designed for the storage of uniformly distributed fundamental memories* $\mathbf{u}^1, \ldots, \mathbf{u}^{160}$ *of length* $N = 100$ *with components in the multivalued set* $\mathcal{S}$ *given by* (1.20) *with* $K = 256$*. The multistate ECNNs have been initialized with a random state* $\mathbf{x}(0)$ *uniformly distributed in* $\mathcal{S}^N$*. Figure* 16 *shows the evolution of the energy of a hyperbolic-valued multistate model using synchronous and asynchronous update modes. In both cases, the hypercomplex-valued ECNN failed to settle at an equilibrium.*

We would like to point out that we can ensure the stability of hyperbolic-valued multistate RCNNs by either considering a different reverse-involution or a different activation

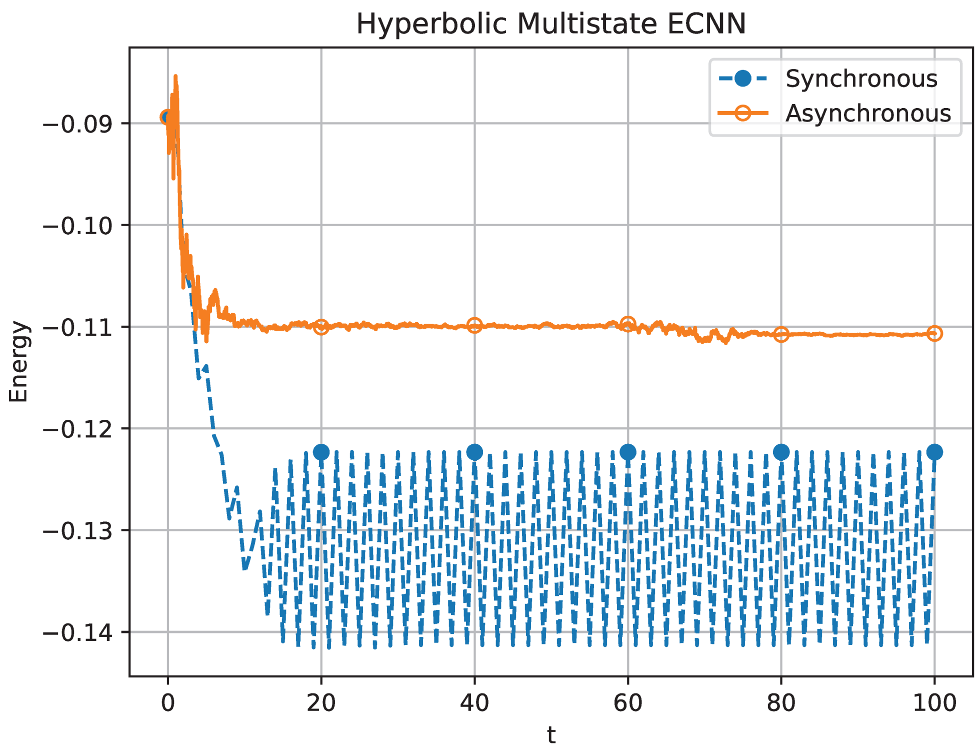


Figure 16 – Evolution of the energy of the hyperbolic-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ with a resolution factor $K = 256$.

function. In fact, by considering the trivial reverse involution $\tau(p) = p$ instead the natural conjugation $\tau(p) = \bar{p}$, the hyperbolic-valued RCNNs coincide with the complex-valued RCNN with the natural conjugation. Alternatively, (3.14) yields a convergent sequence of multivalued vectors if we adopt the function $\overline{\mathrm{csgn}} : \mathcal{D} \to \mathcal{S}$ defined by $\overline{\mathrm{csgn}}(p) = \mathrm{csgn}(\bar{p})$ for all $p \in \mathcal{D}$. In this case, however, the fundamental memories may fail to be stationary states.

Example 3. *We synthesized a hyperbolic-valued ECNN designed for the storage of the fundamental memory set $\mathcal{U}$ given by* (3.30) *using $\alpha = \beta = 1$ and the activation function $\overline{\mathrm{csgn}}$. Figure 17 depicts the entire dynamic of this hyperbolic-valued ECNN using synchronous and asynchronous update mode. In contrast to the model in Example 2, this hyperbolic-valued ECNN always settle at an equilibrium. However, the fundamental memory $\mathbf{u}^1$, which corresponds to the 4th state, is not a stationary state.*

Finally, we would like to recall that both complex and hyperbolic number systems are isomorphic to instances of real Clifford algebras. Apart from complex and hyperbolic-valued modes, Hopfield neural networks on Clifford algebras have been investigated by Vallejo and Bayro-Corrochano (2008), Kuroe and collaborators (Kuroe et al., 2011; Kuroe, 2013), and Kobayashi (2018b). A brief account on the stability of Hopfield neural networks on Clifford algebras of dimension 2 with the so-called split-sign activation function can be found in de Castro

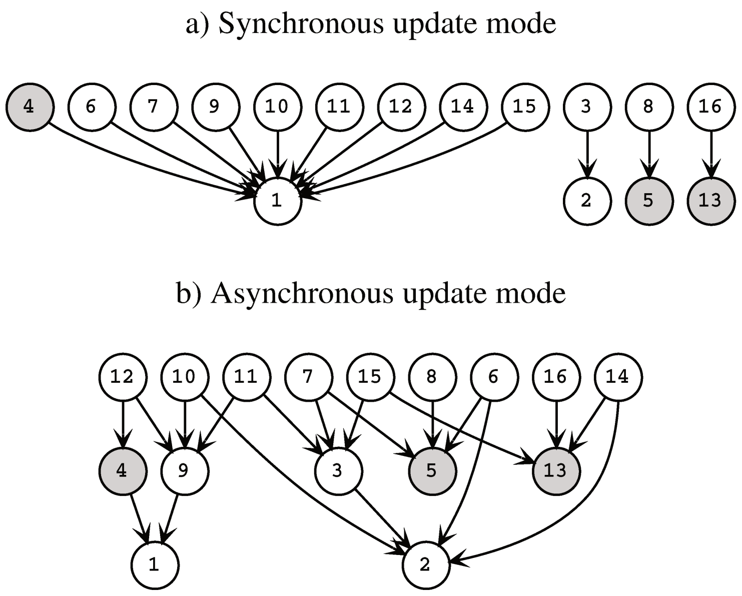


Figure 17 – Dynamic of the hyperbolic-valued ECNN with $\alpha = \beta = 1$ and the activation function $\overline{\text{csgn}}(p) = \text{csgn}(\bar{p})$ using a) synchronous and b) asynchronous update modes.

and Valle (2020). Following the reasoning presented in this section and in de Castro and Valle (2020), we believe one can easily investigate the stability of complex-valued, hyperbolic-valued, and dual number-valued RCNNs with the split-sign activation function.

3.5.2.1 Quaternion-Valued Multistate RCNNs

As a growing and active research area, quaternion-valued neural networks have been successfully applied for signal processing and times series prediction (Arena et al., 1997; Xia et al., 2015; Xu et al., 2016; Papa et al., 2017; Xiaodong et al., 2017; Greenblatt and Agaian, 2018) as well as image processing and classification (Minemoto et al., 2016, 2017; Castro and Valle, 2017a; Chen et al., 2017; Shang and Hirose, 2014). Furthermore, quaternion-valued outperformed their corresponding real-valued models in many of the aforementioned applications.

In the context of recurrent networks and associative memories, research on quaternion-valued Hopfield neural network dates from late 2000s (Isokawa et al., 2007, 2008b,a). Precisely, based on the works of Jankowski et al. (1996), Isokawa et al. introduced a quaternion-valued multisate signum function using quantizations of the phase-angle representation of quaternions (Isokawa et al., 2008a; Bülow, 1999). The quaternion-valued multisate signum function have been used to define multistate Hopfield network which can be synthesized using the Hebbian learning or the projection rule (Isokawa et al., 2008a, 2013). Unfortunately, the quaternion-valued multistate signum function is not a $\mathcal{B}$-function on the quaternions with the natural conjugation de Castro and Valle (2020). Therefore, we cannot ensure the stability of a quaternion-valued

Hopfield network based on the quaternion-valued multistate signum function. In fact, there are simple examples in which this network does not come to rest at an equilibrium (Valle and Castro, 2018). The continuous-valued and the twin-multistate activation functions provide alternatives to define stable quaternion-valued Hopfield neural networks (Valle, 2014b; Valle and Castro, 2018; Kobayashi, 2016b, 2017b).

From the Cauchy-Schwartz inequality, one can show that the continuous-valued activation function σ defined in (2.7) is a $\mathcal{B}$-function in the set of quaternions with the natural conjugation. From Theorem 3.1, and in accordance with Valle (2018), the sequence produced by a quaternion-valued RCNN with the continuous-valued activation function σ is always convergent. The storage capacity and noise tolerance of the continuous-valued quaternionic ECNN have been investigated in Valle (2018). Let us now turn our attention to the new quaternion-valued multistate RCNN obtained by considering the twin-multistate activation function.

The twin-multistate activation function, introduced by Kobayashi (2017a), is defined using the complex-valued multistate signum function given by (1.21). Precisely, from the identity $\mathbf{ij} = \mathbf{k}$, we can write a quaternion as

$$q = (q_0 + q_1\mathbf{i}) + (q_2 + q_3\mathbf{i})\mathbf{j} = z_0 + z_1\mathbf{j}, \tag{3.31}$$

where $z_0 = q_0 + q_1\mathbf{i}$ and $z_1 = q_2 + q_3\mathbf{i}$ are complex numbers. Given a resolution factor K, the twin-multistate activation function is defined by means of the equation

$$\text{tsgn}(q) = \text{csgn}(z_0) + \text{csgn}(z_1)\mathbf{j}, \tag{3.32}$$

for all $q = z_0 + z_1\mathbf{j}$ such that $z_0, z_1 \in \mathcal{D}$, where $\mathcal{D}$ is the domain of the complex-valued multistate signum function defined by (1.19). Note that the domain $\mathcal{D}_t$ and codomain $\mathcal{S}_t$ of the twin-multistate activation function tsgn are respectively the subset of quaternions

$$\mathcal{D}_t = \{q = z_0 + z_1\mathbf{j} : z_0, z_1 \in \mathcal{D}\}, \tag{3.33}$$

and

$$\mathcal{S}_t = \{s = w_0 + w_1\mathbf{j} : w_0, w_1 \in \mathcal{S}\}, \tag{3.34}$$

where $\mathcal{S}$ is the set of complex numbers defined by (1.20). It is not hard to show that $\mathcal{S}_t$ is compact. Furthermore, tsgn is a $\mathcal{B}$-function on the quaternions with the natural conjugation. In fact, using (3.31), we can express the quaternion-valued symmetric bilinear form $\mathcal{B}_{\mathbb{Q}}$ as the sum of the complex-valued symmetric bilinear form $\mathcal{B}_{\mathbb{C}}$ as follows for any quaternions $q = z_0 + z_1\mathbf{j}$ and $p = w_0 + w_1\mathbf{j}$:

$$\mathcal{B}_{\mathbb{Q}}(q, s) = \mathcal{B}_{\mathbb{C}}(z_0, w_0) + \mathcal{B}_{\mathbb{C}}(z_1, w_1). \tag{3.35}$$

Since csgn is a $\mathcal{B}$-function on the complex-numbers with the natural conjugation, we obatin the following inequality for all $q = z_0 + z_1\mathbf{j} \in \mathcal{D}_t$ and $s = w_0 + w_1\mathbf{j} \in \mathcal{S}_t \backslash \{\text{tsgn}(q)\}$:

$$\mathcal{B}_{\mathbb{Q}}(\text{tsgn}(q), q) = \mathcal{B}_{\mathbb{C}}(\text{csgn}(z_0), z_0) + \mathcal{B}_{\mathbb{C}}(\text{csgn}(z_1), z_1) \tag{3.36}$$

$$> \mathcal{B}_{\mathbb{C}}(w_0, z_0) + \mathcal{B}_{\mathbb{C}}(w_1, z_1) \tag{3.37}$$

$$= \mathcal{B}_{\mathbb{Q}}(s, q). \tag{3.38}$$

From Theorem 3.1, we conclude that quaternion-valued RCNNs with twin-multistate activation functions always settle at a stationary state. The following example and computational experiment confirm this remark.

Example 4. *Using a resolution factor $K = 4$, we obtain the set*

$$\mathcal{S}_t = \{\pm 1 \pm \mathbf{j}, \pm 1 \pm \mathbf{k}, \pm \mathbf{i} \pm \mathbf{j}, \pm \mathbf{i} \pm \mathbf{k}\}. \tag{3.39}$$

Let us consider the fundamental memory set

$$\mathcal{U} = \{\mathbf{u}^1 = 1 - \mathbf{k}, \quad \mathbf{u}^2 = \mathbf{i} + \mathbf{j}, \quad \mathbf{u}^3 = -\mathbf{i} + \mathbf{j}\}. \tag{3.40}$$

We synthesized the quaternion-valued multistate ECNN designed for the storage and recall of the fundamental memory set $\mathcal{U}$ given by (3.40)*. Like the previous experiment, we used $\alpha = 1$ and $\beta = 1$. In this simple example the update mode is irrelevant because the network has a single state neuron. Moreover, the dynamic of this quaternion-valued multistate ECNN is identical to the sychronous complex-valued multistate neural network depicted on Figure 13a).*

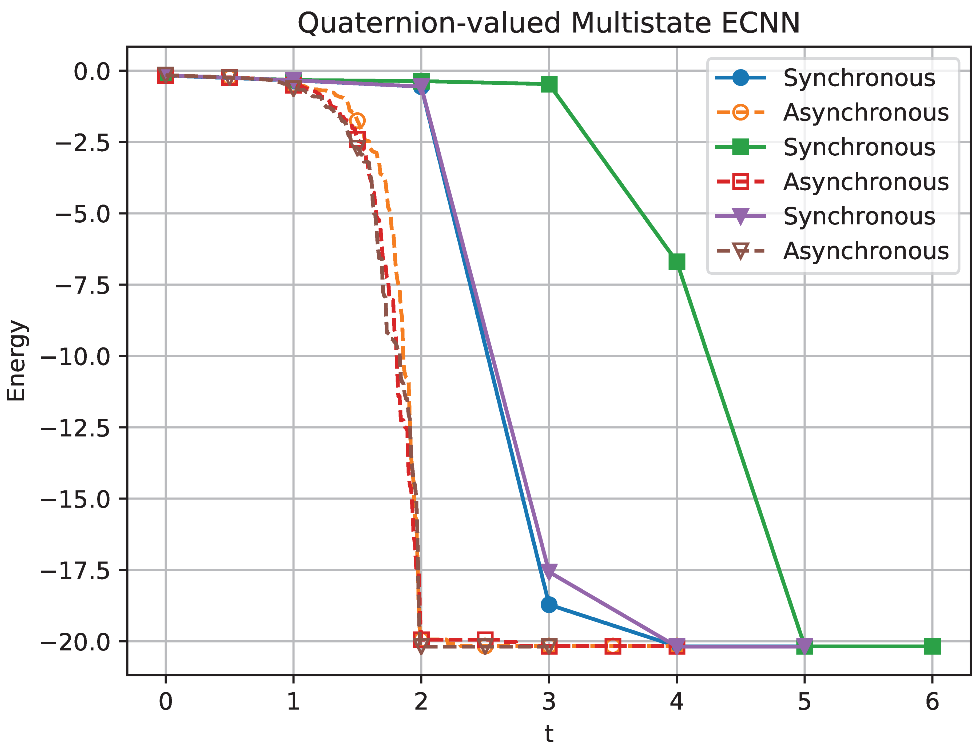


Figure 18 – Evolution of the energy of the quaternion-valued multistate ECNN designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$ and $K = 16$. The curves marked by "●" and "○", "■" and "□", as well as "▼" and "▽" refer to pairs of ECNNs that differ only from the update mode.

Experiment 3. *We synthesized a quaternion-valued multistate ECNN designed for the storage and recall of randomly generated fundamental memory sets $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \subset \mathcal{D}_t^N$, where*

$N = 100$ and $P = 160$. In this experiment, we considered $\alpha = 10/(2N)$, $\beta = e^{-10}$, and $K = 16$. Furthermore, since the set $\mathcal{D}_t$ is dense on $\mathbb{Q}$, we refrained to check the condition $a_i(t) \in \mathcal{D}_t$ in our computational implementation. Figure 18 shows the evolution of the energy of the quaternion-valued multistate ECNN using synchronous and asynchronous updates. Note that the energy always decreases until the networks settle at a stationary state. Like the previous hypercomplex-valued ECNN models, the computational implementation of the synchronous quaternion-valued multistate ECNN is usually faster than the asynchronous version.

3.5.2.2 Hypercomplex-valued RCNNs on Cayley-Dickson Algebras

Complex numbers, quaternions, and octonions are instance of Cayley-Dickson algebras. Cayley-Dickson algebras are hypercomplex number systems of dimension 2^k in which the product and the conjugation (reverse involution) are defined recursively as follows: First, let A_0 denote the real number system. Given a Cayley-Dickson algebra A_k, the next algebra A_{k+1} comprises all pairs $(x, y) \in A_k \times A_k$ with the component-wise addition, the conjugation given by

$$\overline{(x, y)} = (\bar{x}, -y), \tag{3.41}$$

and the product defined by

$$(x_1, y_1)(x_2, y_2) = (x_1x_2 - y_2\bar{y}_1, \bar{x}_1y_2 + x_2y_1), \tag{3.42}$$

for all $(x_1, y_1), (x_2, y_2) \in A_{k+1}$. The Cayley-Dickson algebras A_1, A_2, and A_3 coincide respectively with the complex numbers, quaternions, and octonions. The symmetric bilinear form $\mathcal{B} : A_k \times A_k \to \mathbb{R}$ is given by

$$\mathcal{B}(p, q) = \sum_{j=1}^{n} p_jq_j, \tag{3.43}$$

for all $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n \in A_k$ and $q = q_0 + q_1\mathbf{i}_1 + \ldots + q_n\mathbf{i}_n \in A_k$, where $n = 2^k - 1$. Note that the symmetric bilinear form $\mathcal{B}$ given by (3.43) corresponds to the inner product between $p \equiv (p_0, \ldots, p_n)$ and $q \equiv (q_0, \ldots, q_n)$. As a consequence, Cayley-Dickson algebras enjoy many properties from Euclidean spaces, including the Cauchy-Schwarz inequality. On the downside, the Cayley-Dickson algebras are non-commutative for $k \geqslant 2$. Thus, particular attention must be given to the order of the terms in (3.42). Furthermore, Cayley-Dickson algebras are non-associative for $k \geqslant 3$. As a consequence, in contrast to complex numbers and quaternions, the product defined by (3.42) cannot be represented by matrix operations for $k \geqslant 3$.

The continuous-valued function σ given by (2.7) is a $\mathcal{B}$-function in any Cayley-Dickson algebra A_k, with $|p| = \sum_{j=1}^{n} p_j^2$ for all $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n$, $n = 2^k - 1$. From Theorem 3.1, hypercomplex-valued RCNNs with the continuous-valued function σ always settle at an equilibrium synchronously as well as asynchronously. This result generalizes the stability analysis of the complex-valued and quaternion-valued RCNNs introduced in Valle's works (Valle, 2014a, 2018).

Apart from the continuous-valued RCNNs, the split sign function is also a $\mathcal{B}$-function in Cayley-Dickson algebras (de Castro and Valle, 2020). The split sign function $\operatorname{sgn} : \mathcal{D}_s \to \mathcal{S}_s$ is defined by means of the following equation for all $p = p_0 + p_1\mathbf{i}_1 + \ldots + p_n\mathbf{i}_n \in A_k$:

$$\operatorname{sgn}(p) = \operatorname{sgn}(p_0) + \operatorname{sgn}(p_1)\mathbf{i}_1 + \ldots + \operatorname{sgn}(p_n)\mathbf{i}_n, \tag{3.44}$$

where

$$\mathcal{D}_s = \{p \in A_k : p_0 p_1 \ldots p_n \neq 0\}, \tag{3.45}$$

and

$$\mathcal{S}_s = \{p \in A_k : p_j \in \{-1, +1\}, \forall j = 0, 1, \ldots, n\}. \tag{3.46}$$

From Theorem 3.1, the sequence produced by hypercomplex-valued RCNNs with $\phi = \operatorname{sgn}$ are all convergent using synchronous and asynchronous update mode. The following experiment confirms this remark.

Experiment 4. *As an illustrative example of a hypercomplex-valued RCNN on Cayley-Dickson algebra, let us consider an octonion-valued ECNN with the split sign activation function. Precisely, we synthesized octonion-valued ECNNs designed for the storage and recall of fundamental memory sets* $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\} \in \mathcal{S}_s^N$*, where* $\mathcal{S}_s$ *is given by* (3.46)*, with* $N = 100$ *and* $P = 160$*. In this experiment, we adopted the parameters* $\alpha = 10/(8N)$ *and* $\beta = e^{-10}$*. The fundamental memories as well as the initial states have been generated using an uniform distribution, that is, the entries* $u_i^\xi = u_{i0}^\xi + u_{i1}^\xi\mathbf{i}_1 + \ldots + u_{i7}^\xi\mathbf{i}_7$ *of the fundamental memories are such that* $\Pr[u_{ij}^\xi = -1] = \Pr[u_{ij}^\xi = +1] = 1/2$ *for all* $\xi = 1, \ldots, P$*,* $i = 1, \ldots, N$*, and* $j = 0, 1, \ldots, 7$*. Figure 19 shows the evolution of the energy of some octonion-valued ECNN models using both synchronous and asynchronous update. As expected, using both update modes, the energy decreases until the network settles at a stationary state.*

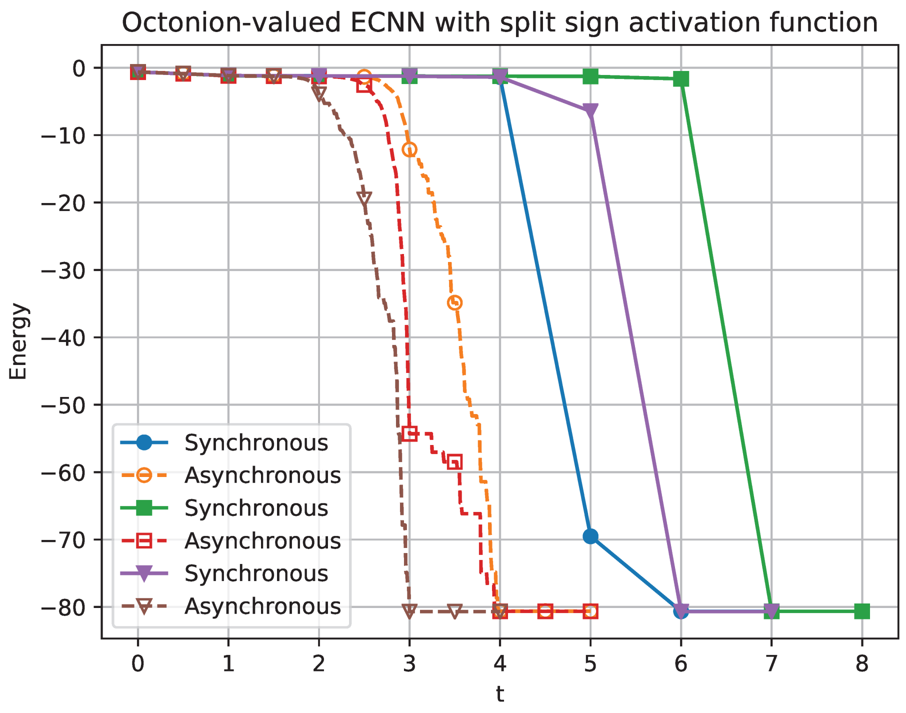


Figure 19 – Evolution of the energy of the octonion-valued ECNN with the split sign function designed for the storage of $P = 160$ uniformly generated fundamental memories of length $N = 100$. The curves marked by "●" and "○", "■" and "□", as well as "▼" and "▽" refer to pairs of ECNNs that differ only from the update mode.

3.5.3 HRCNNs Image Experiments

In the previous section, we provided several examples of hypercomplex-valued ECNNs and, by means of simple computational experiments, we validated Theorem 3.1. Let us now compare the noise tolerance of the hypercomplex-valued ECNNs designed for the storage and recall of gray-scale images.

3.5.3.1 Storage and Recall of Gray-Scale Images

The CIFAR dataset contain thousands color images of size 32×32. We randomly selected $P = 200$ images from the CIFAR dataset, converted them to gray-scale images, and encoded them in an appropriate manner as hypercomplex-valued vectors. Precisely, we encoded the gray-scale images into hypercomplex-valued vectors as follows where $x \in \{0, 1, \ldots, 255\}$ denotes a gray-scale pixel value:

- *Bipolar:* Using the binary representation $b_1 \ldots b_8$, a gray-scale pixel value x can be associated to an array $[2b_1 - 1, \ldots, 2b_8 - 1] \in \{-1, +1\}^8$. Therefore, a gray-scale image of size 32×32 can be converted into a bipolar vector of length $N_b = 8192$ by concatenating all 8-dimensional bipolar arrays.

- *Complex-valued multistate:* Using a resolution factor $K = 256$, a gray-scale value $x \in \{0, \ldots, 255\}$ yields a complex number $z \in \mathcal{S}$, where $\mathcal{S}$ is the set given by (1.20), by means of the equation $z = e^{2\pi x \mathbf{i}/K}$. As a consequence, a gray-scale image of size 32×32 corresponds to a complex-valued vector in $\mathcal{S}^{N_c}$, where $N_c = 1024$.

- *Quaternion-valued multistate:* The binary representation $b_1 \ldots b_8$ of $x \in \{0, 1, \ldots, 255\}$ allow us to define integers $x_1 = 8b_4 + 4b_3 + 2b_2 + b_1$ and $x_2 = 8b_8 + 4b_7 + 2b_6 + b_5$. Using a resolution factor $K = 16$, the equation $q = e^{2\pi x_1 \mathbf{i}/K} + e^{2\pi x_2 \mathbf{i}/K}\mathbf{j}$ yields a quaternion in the set $\mathcal{S}_t$ given by (3.34). In this way, a gray-scale image of size 32×32 corresponds to a quaternion-valued vector in $\mathcal{S}_t^{N_q}$, where $N_q = 1024$.

- *Octonion-valued:* Using the binary representation $b_1 \ldots b_8$ of an integer $x \in \{0, \ldots, 255\}$, the equation $p = (2b_1 - 1) + (2b_2 - 1)\mathbf{i}_1 + \ldots + (2b_8 - 1)\mathbf{i}_7$ yields an octonion $p \in \mathcal{S}_s$, where $\mathcal{S}_s$ is given by (3.46). As a consequence, a gray-scale image of size 32×32 corresponds to an octonion-valued vector in $\mathcal{S}_s^{N_o}$, where $N_o = 1024$.

The $P = 200$ hypercomplex-valued vectors have been stored in bipolar, complex-valued, quaternion-valued, and octonion-valued ECNN models. The parameters used in this experiment can be found in Table 3. Furthermore, we introduced Gaussian noise into one of the selected images and, in a similar fashion, we converted the corrupted image into a hypercomplex-valued vector which have been fed as input to the hypercomplex-valued ECNNs. Figure 20 depicts a gray-scale image from the CIFAR dataset, its corresponding version corrupted by

Table 3 – Parameters of the hypercomplex-valued ECNN models designed for the storage and recall of CIFAR images where $N_b = 8192$ and $N_c = N_q = N_o = 1024$.

	Bipolar	Complex	Quaternion	Octonion
α	$20/(N_b)$	$20/N_c$	$20/(2N_q)$	$20/(8N_o)$
β	e^{-20}	e^{-20}	e^{-20}	e^{-20}
K	—	256	16	—

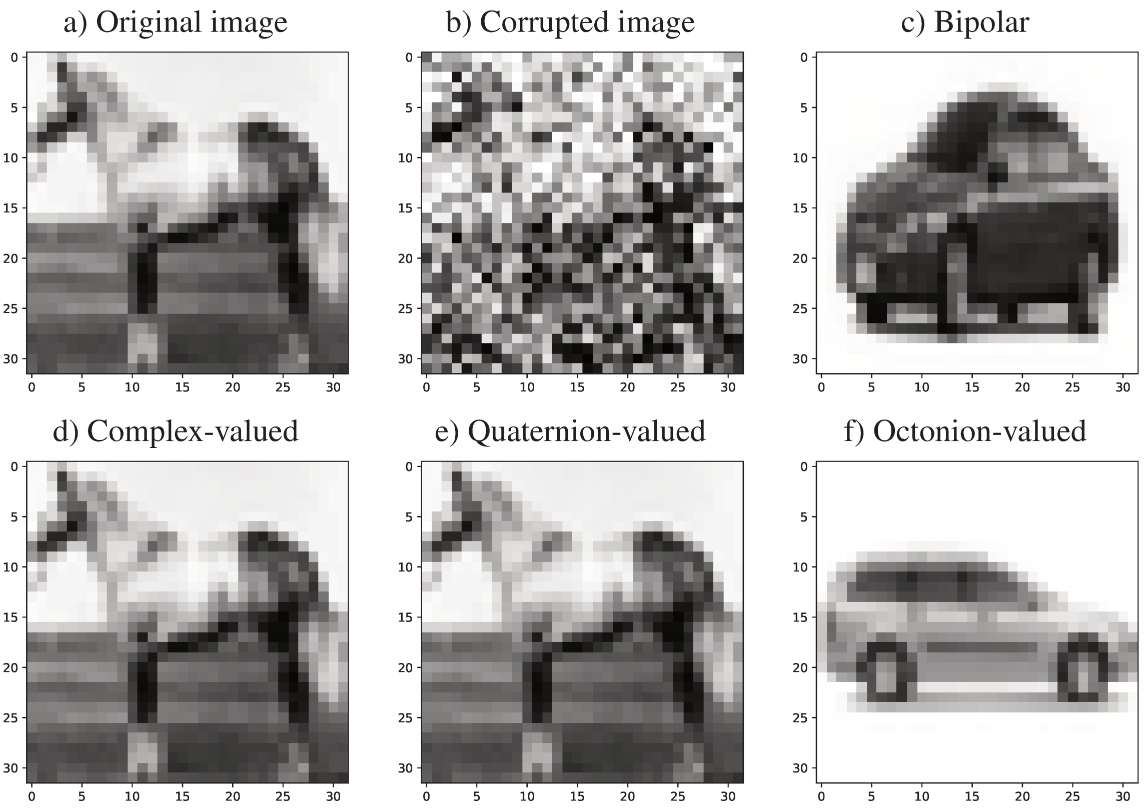


Figure 20 – a) Original (undistorted) gray-scale image, b) input image corrupted by Gaussian noise with standard variation 100, images retrieved by c) bipolar ECNN, d) complex-valued multistate ECNN with $K = 246$, e) quaternion-valued multistate ECNN with $K = 16$, and split-sign octonion-valued ECNN using asynchronous update.

Gaussian noise with standard variation 100, and the images retrieved by the hypercomplex-valued ECNNs using asynchronous update. Note that the complex-valued and quaternion-valued multistate ECNN succeeded in the retrieval of the original horse image. In contrast, the bipolar and octonion-valued models produced a different image as output; they failed to retrieve the undistorted horse image.

Let us now compare the noise tolerance of the four synchronous hypercomplex-valued ECNNs designed for the storage and recall of images from the CIFAR dataset. To this end, we randomly selected $P = 200$ images and corrupted one of them with Gaussian noise with several different standard variation. We repeated the experiment 100 times and computed the probability of successful recall of the original image. Figure 21 depicts the outcome of

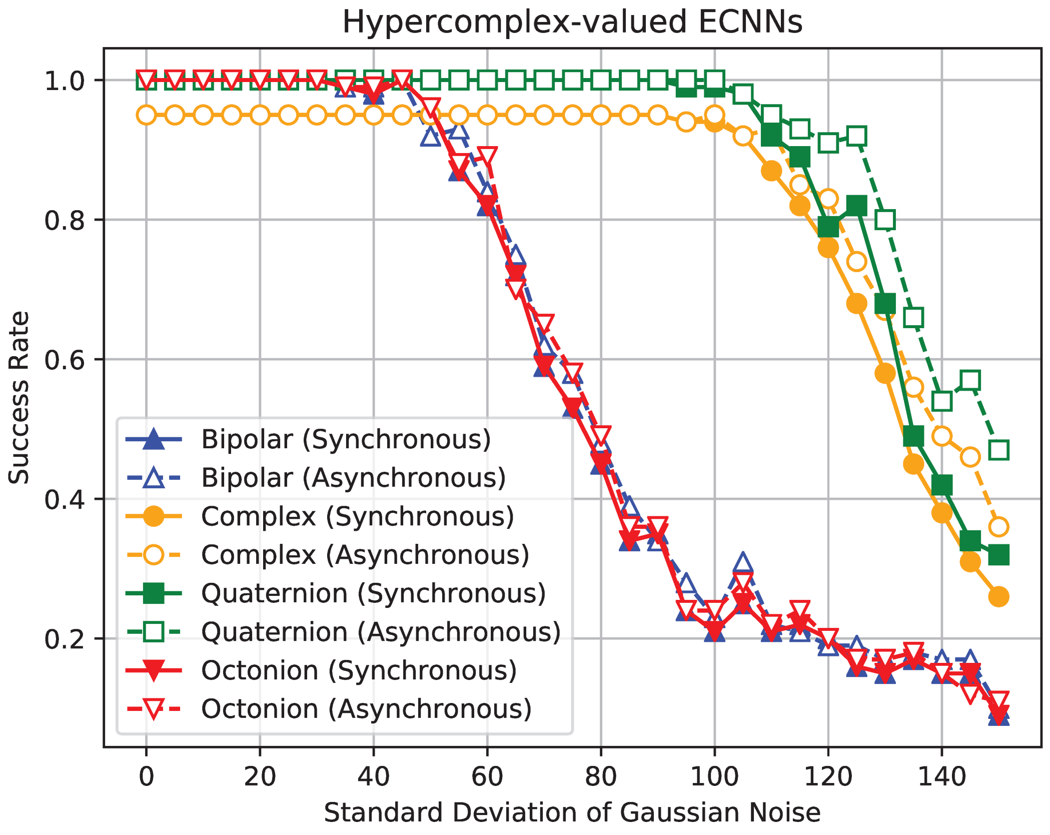


Figure 21 – Probability of successful recall by the standard variation of Gaussian noise introduced into the input image.

this experiment using both synchronous and asynchronous update. Note that the quaternion-valued model yielded the largest noise tolerance among the four hypercomplex-valued ECNNs. Furthermore, the asynchrounous update yielded a probability of successful recall larger than the synchronous update. The second largest noise tolerance was produced by the complex-valued ECNN model using asynchronous update. Finally, the bipolar and the octonion-valued ECNN coincide using synchronous update. Also, there is no significant difference between the bipolar and octonion-valued models synchronous and asynchronous update modes. Concluding, this experiment reveals the potential application of the new hypercomplex-valued ECNNs as associative memories designed for the storage and recall of gray-scale images.

3.5.4 HRPNNs synthetic experiments

Example 5. *In Figure 22 we show the dynamical behavior of the complex-valued multistate EPNN with $\alpha = \beta = 1$, considering the fundamental memory set $\mathcal{U}$ given by (3.30), where $\mathcal{S} = \{1, \mathbf{i}, -1, -\mathbf{i}\}$ is the set given by (1.20). Note that, all the fundamental memories are fixed points of the model, both in synchronous and asynchronous update mode. However, this model has more spurious memories (9th and 16th states) than the complex-valued multistate RCNN* *under the same conditions (see Figure 13).*

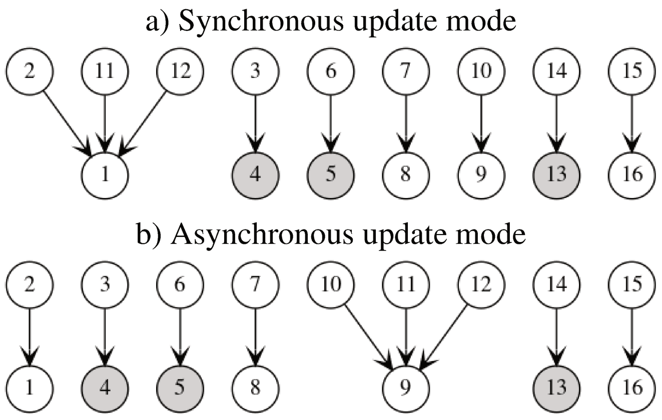


Figure 22 – Dynamic of the complex-valued multistate EPNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes.

Example 6. *Considering the fundamental memory set $\mathcal{U}$ given by (3.30), where $\mathcal{S} = \{1, \mathbf{i}, -1, -\mathbf{i}\}$ is the set given by (1.20) and* $\mathbf{i}$ *denotes the hyperbolic unit, i.e.,* $\mathbf{i}^2 = 1$*. We synthesized hyperbolic-valued mustistate EPNN models designed for the storage of the fundamental memory set $\mathcal{U}$, using $\alpha = 1$ and $\beta = 1$. In this case, using either synchronous or asynchronous update, the model does not cycle between the 5th and 13th states, in comparison with the hyperbolic complex-valued ECNN whose dynamics is shown in Figure 15. An initial explanation for this behavior is the better choice of weights of the output layer of the EPNN model.*

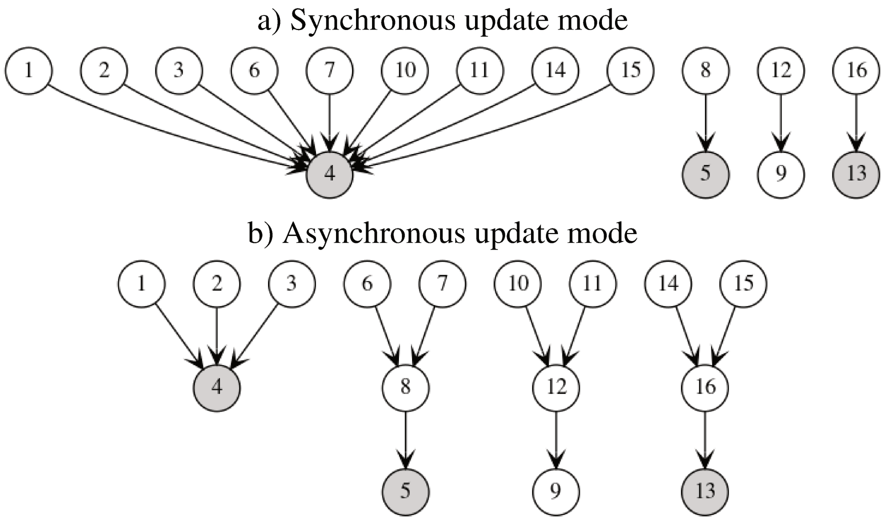


Figure 23 – Dynamic of the hyperbolic-valued multistate RPNN with $\alpha = \beta = 1$ using a) synchronous and b) asynchronous update modes.

Example 7. *We synthesize a hyperbolic-valued EPNN designed for the storage of the fundamental memory set $\mathcal{U}$ given by (3.30) using $\alpha = \beta = 1$ and the activation function $\overline{\text{csgn}}$. Figure 24 depicts the entire dynamic of the hyperbolic-valued EPNN using synchronous and asynchronous* *update mode. In contrast to the model in Examples 2 and 6 , the hyperbolic-valued EPNN always settle at an equilibrium state in asynchronous update mode, creating one spurious memory (9th state). In the synchronous case, the model fails storing all the items and may also cycle between the 5th and 13th states.*

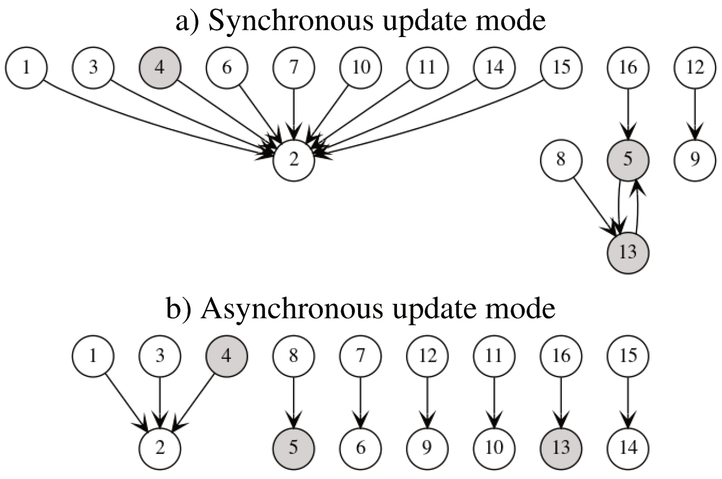


Figure 24 – Dynamic of the hyperbolic-valued EPNN with $\alpha = \beta = 1$ and the activation function $\overline{\text{csgn}}(p) = \text{csgn}(\bar{p})$ using a) synchronous and b) asynchronous update modes.

Moreover, we performed different ECNN and EPNN models storing the number of iterations until they converge. We plot the number of iterations to observe the distributions of convergence velocity for each model.

Example 8. *We synthesize complex-valued multistate EPNN and ECNN models for the storage and recall of fundamental memories $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, where $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$ correspondingly to the α values. Furthermore, we adopt a resolution factor $K = 256$ and randomly generate the components of a fundamental memory. The corrupted input vector is obtained by replacing, with probability $\texttt{pr} \in [0.2; 0.9]$, a component of a fundamental memory $\mathbf{u}^1$ by an element from $\mathcal{S}$ taken at random with uniform probability. We repeat the simulation 50 times for each model. Figure 25 shows the histogram of the number of iterations performed by the models until convergence is reached. The EPNN model has converged in an average of 7.59 iterations and the ECNN an average of 10.52 iterations in this particular experiment. The distribution for EPNN model is concentrated in lower values than the ECNN model, this means that EPNN tend to converge faster than ECNN for different parameter conditions and dimensions.*

Example 9. *Analogously to the complex-valued case, we synthesize quaternion-valued multistate EPNN and ECNN models for the storage and recall of fundamental memories $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$, where $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$ correspondingly to the α values. Besides, we adopt a resolution factor $K = 16$ and randomly generate the components of a fundamental memory. The corrupted*

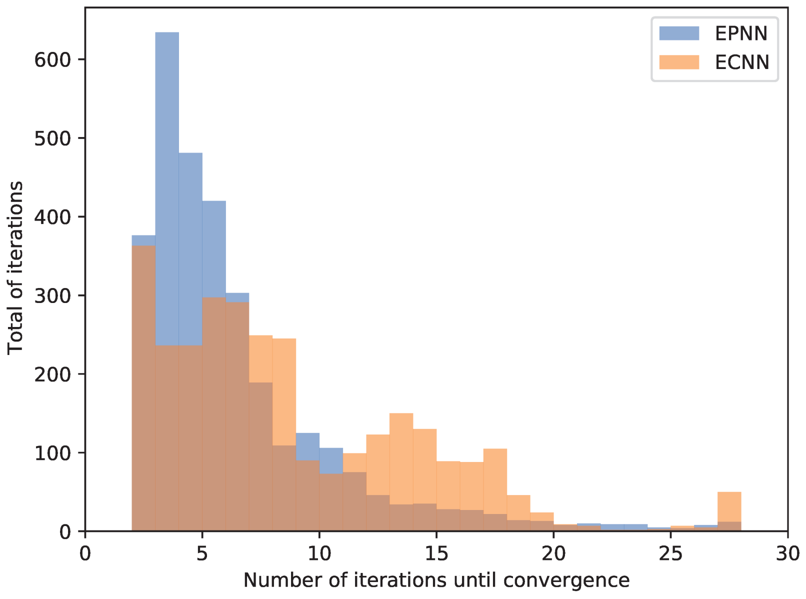


Figure 25 – Histogram of the numbers of iterations for complex-valued multi-state ECNN and EPNN models in synchronous update mode, using $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$, and a resolution factor $K = 256$. The corrupted input vector is obtained by replacing, with probability $\mathtt{pr} \in [0.2; 0.9]$, a component of a fundamental memory $\mathbf{u}^1$ by an element from $\mathcal{S}$ taken at random with uniform probability.

input vector is obtained by replacing, with probability $\mathtt{pr} \in [0.2; 0.9]$*, a component of a fundamental memory* $\mathbf{u}^1$ *by an element from* $\mathcal{S}$ *taken at random with uniform probability. We repeat the simulations totalizing 50 iteration numbers for each model. Figure 26 shows the distribution of the number of iterations performed by the models until convergence is reached. The distribution shows that the EPNN model tends to converge faster than the ECNN model under the same parameter conditions. Moreover, the EPNN model has converged in an average of 3.63 iterations and the ECNN converged in an average of 5.75 iterations in this particular experiment.*

3.5.5 HRPNNs Image Experiments

In Section 3.5.3, we addressed the noise tolerance of the hypercomplex-valued ECNNs designed for the storage and recall of gray-scale images. Now, we repeat the same experiment using the HRPNNs. Figure 27 depicts a gray-scale image from the CIFAR dataset, its corresponding version corrupted by Gaussian noise with a standard variation of 100. For each hypercomplex-valued EPNNs, we used asynchronous update mode. Note that the complex-valued, quaternion-valued, and octonion multistate EPNN succeeded in retrieving the original horse image. In contrast, the bipolar model produced a different image as output, failing to retrieve the undistorted horse image.

Analogously to the ECNN, we store $P = 200$ hypercomplex-valued vectors in

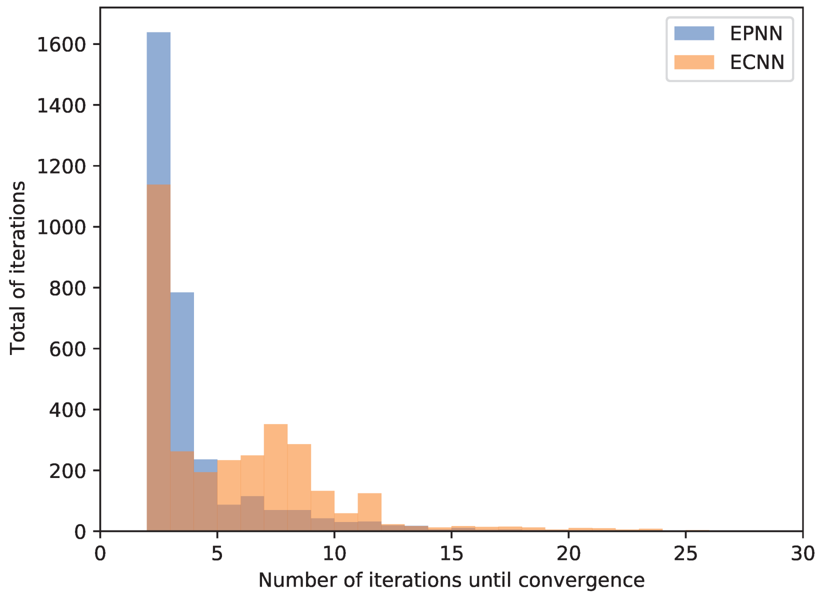


Figure 26 – Distribution of the numbers of iterations for the quaternion-valued multi-state ECNN and EPNN models in synchronous update mode, using $N = 200$ and $P \in \{25, 50, 100, 200\}$, using $\alpha = 1/N, 2/N, 5/N, 10/N$ and $\beta = e^{-1}, e^{-2}, e^{-5}, e^{-10}$, and resolution factor $K = 16$. The corrupted input vector is obtained by replacing, with probability $\texttt{pr} \in [0.2; 0.9]$, a component of a fundamental memory $\mathbf{u}^1$ by an element from $\mathcal{S}$ taken at random with uniform probability.

bipolar, complex-valued, quaternion-valued, and octonion-valued EPNN models. The parameters used in this experiment can also be found in Table 3. Figure 28 depicts the outcome of this experiment using only synchronous update mode. Note that the quaternion-valued model yielded the most considerable noise tolerance among the four hypercomplex-valued EPNNs. It also outperformed the ECNN models in synchronous update mode. Moreover, the complex-valued and the octonion-valued EPNN are superior to their corresponding ECNN. There is no significant difference between the bipolar model and the octonion-valued ECNN, all operating in synchronous update mode. Concluding, this experiment reveals the potential application of the new hypercomplex-valued EPNN as associative memories designed for the storage and recall of gray-scale images.

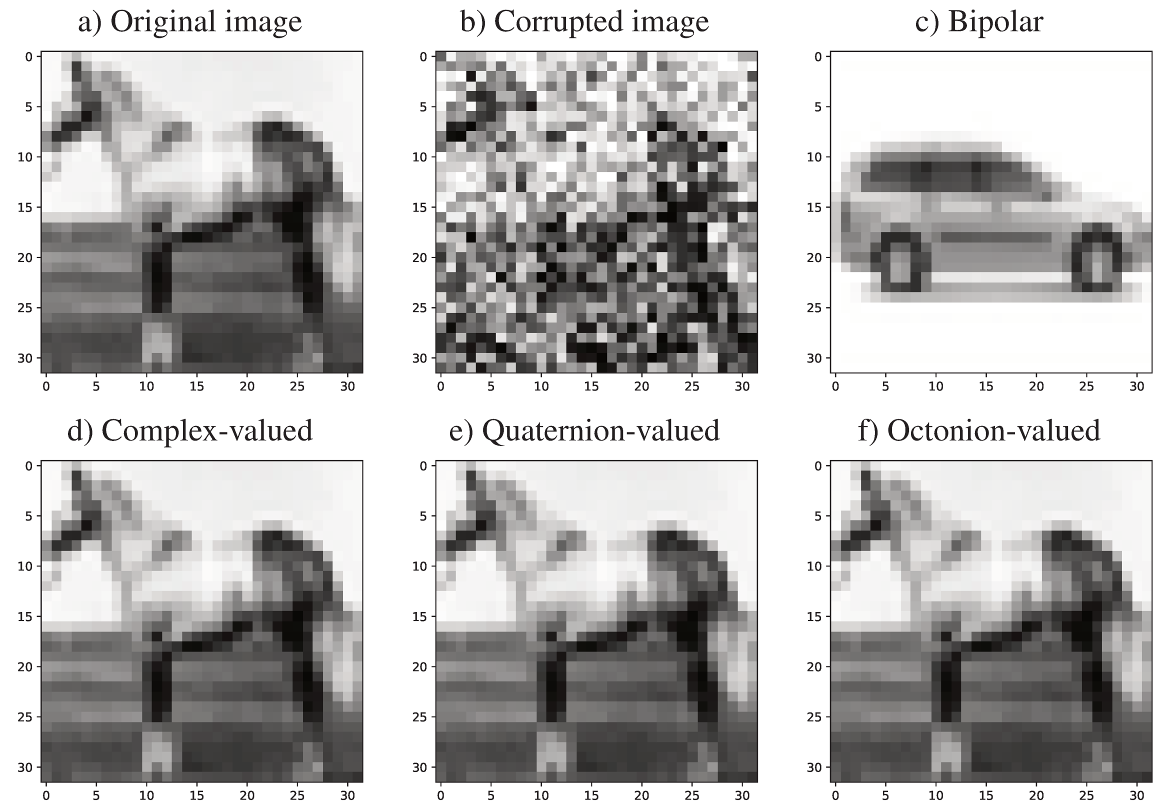


Figure 27 – a) Original (undistorted) gray-scale image, b) input image corrupted by Gaussian noise with standard variation 100, retrieved images by c) bipolar EPNN, d) complex-valued multistate EPNN with $K = 246$, e) quaternion-valued multistate EPNN with $K = 16$, and split-sign octonion-valued EPNN using asynchronous update.

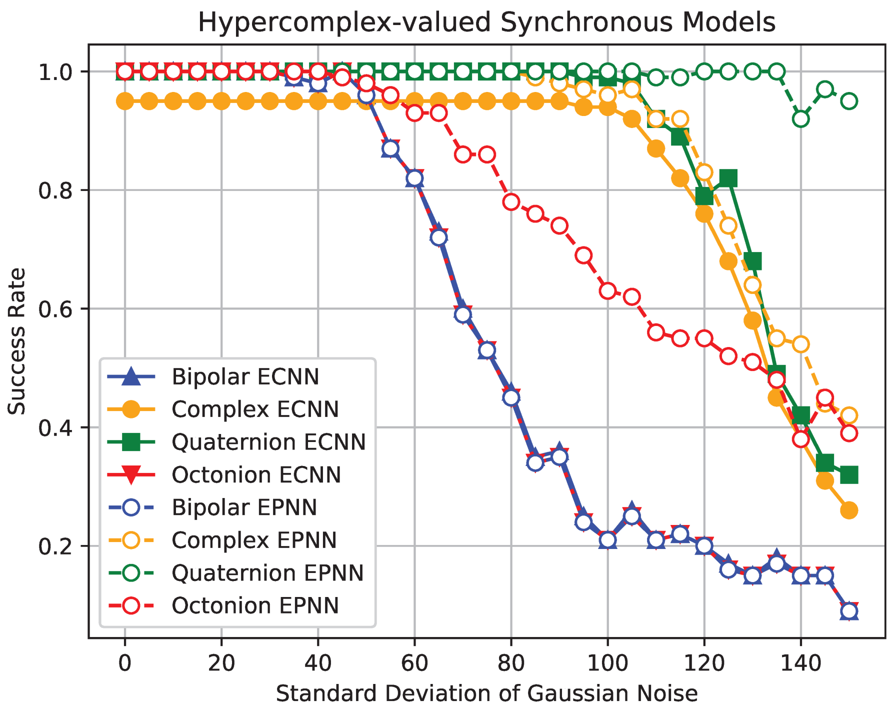


Figure 28 – Probability of successful recall by the standard variation of Gaussian noise introduced into the input image, using ECNN and EPNN models.

4 Ensemble of Classifiers Combined Using RCNNs

An ensemble method should cleverly combine a group of base classifiers to yield an improved classifier. The majority vote is an example of a methodology used to combine classifiers in an ensemble method. In this chapter, we propose a new way to combine classifiers using an associative memory model. Precisely, we introduce ensemble methods based on recurrent correlation associative memories (RCNNs) for binary and multi-class classification problems. For the binary case, we show that an RCNN-based ensemble classifier can be viewed as a majority vote classifier whose weights depend on the similarity between the base classifiers and the resulting ensemble method. Moreover, the RCNN-based ensemble combines the classifiers using a recurrent scheme. Computational experiments confirm the potential application of the RCNN-based ensemble method for binary classification problems. Finally, for multi-class classification problems, we use real, complex, and quaternion-valued RCNN models, obtaining comparable but lower performance than the methods from the literature. We discuss the results and the possible issues related to low performance of the HRCNN-based enemble classifier.

4.1 Introduction

Inspired by the idea that multiple opinions are crucial before making a final decision, ensemble methods make predictions by consulting multiple different predictors (Ponti Jr, 2011). Apart from their similarity with some natural decision-making methodologies, ensemble methods have a strong statistical background. Namely, ensemble methods aim to reduce the variance – thus increasing the accuracy – by combining multiple predictors. Due to their versatility and effectiveness, ensemble methods have been successfully applied to a wide range of problems including classification, regression, and feature selection.

Although there is no rigorous definition, an ensemble classifier can be conceived as a group of base classifiers, also called weak classifiers (Kuncheva, 2014). As to the construction of an ensemble classifier, we must take into account the diversity of the base classifiers and the rule used to combine them (Kuncheva, 2014; Polikar, 2012). There are a plethora of ensemble methods in the literature, including bagging, pasting, random subspace, boosting, and stacking (Breiman, 1996; Ho, 1998; Zhang and Ma, 2012; Géron, 2019). For example, a bagging ensemble classifier is obtained by training copies of a single base classifier using different subsets of the training set (Breiman, 1996). Similarly, a random subspace classifier is obtained by training copies of a classifier using different subsets of features (Ho, 1998). In both bagging and random subspace ensembles, the base classifiers are then combined using a voting scheme. Random forest is a successful example of an ensemble of decision tree classifiers trained using both bagging and random subspace ensemble ideas (Breiman, 2001).

In contrast to the traditional majority voting, we propose to combine the base classifiers using an associative memory. At this point, we would like to remark that associative memories have previously been used by Kultur et al. (2009) to improve the performance of an ensemble method. Apart from addressing a regression problem, Kultur et al. use an associative memory in parallel to an ensemble of multi-layer perceptrons. The resulting model is called *ensemble of neural networks with associative memory (ENNA)*. Our approach, in contrast, uses an associative memory to combine the base classifiers. Besides, Kultur et al. (2009) associate patterns using the k-nearest neighbor algorithm which is formally a non-parametric method used for classification or regression. Differently, we use recurrent correlation associative memories, which are models conceived to implement associative memories.

4.2 Ensemble of Binary Classifiers

An ensemble classifier combines a group of single classifiers to provide better classification accuracy than a single one (Ponti Jr, 2011; Zhang and Ma, 2012; Kuncheva, 2014). Although this approach is partially inspired by the idea that multiple opinions are crucial before making a final decision, ensemble classifiers have a strong statistical background. In fact, ensemble classifiers reduce the variance combining the base classifiers. Furthermore, when the amount of training data available is too small compared to the size of the hypothesis space, the ensemble classifier mixes the base classifiers reducing the risk of choosing the wrong single classifier (Kittler and Roli, 2003).

Formally, let $\mathcal{T} = \{(\boldsymbol{t}_1, d_1), \ldots, (\boldsymbol{t}_M, d_M)\}$ be a training set where $\boldsymbol{t}_i \in \mathcal{X}$ and $d_i \in \mathcal{C}$ are respectively the feature sample and the class label of the ith training pair. Here, $\mathcal{X}$ denotes the feature space and $\mathcal{C}$ represents the set of all class labels. In a binary classification problem, we can identify $\mathcal{C}$ with $\mathbb{B} = \{-1, +1\}$. Moreover, let $h_1, h_2, \ldots, h_P : \mathcal{X} \to \mathcal{C}$ be base classifiers trained using the whole or part of the training set $\mathcal{T}$.

Usually, the base classifiers are chosen according to their accuracy and diversity. On the one hand, an accurate classifier is one that has a better error rate than random guessing on new instances. On the other hand, two classifiers are diverse if they make different errors on new instances (Hansen and Salamon, 1990; Kittler and Roli, 2003).

Bagging and random subspace ensembles are examples of techniques that can be used to ensure the diversity of the base classifiers. The idea of bagging, an acronym for *Bootstrap AGGregatING*, is to train copies of a certain classifier h on subsets of the training set $\mathcal{T}$ (Breiman, 1996). The subsets are obtained by sampling the training $\mathcal{T}$ with replacement, a methodology known as *bootstrap sampling* (Kuncheva, 2014). In a similar fashion, random subspace ensembles are obtained by training copies of a certain classifier h using different subsets of the feature space

(Ho, 1998). Random forest, which is defined as an ensemble of decision tree classifiers, is an example of an ensemble classifier that combines both bagging and random subspace techniques (Breiman, 2001).

Another important issue that must be addressed in the design of an ensemble classifier is how to combine the base classifiers. In the following, we review the majority voting methodology – one of the oldest and widely used combination scheme. The methodology based on associative memories is introduced and discussed subsequently.

4.2.1 Majority Voting Classifier

As remarked by Kuncheva (2014), majority voting is one of the oldest strategies for decision making. In a wide sense, a majority voting classifier yields the class label with the highest number of occurrences among the base classifiers (Van Erp et al., 2002; Géron, 2019).

Formally, let $h_1, h_2, \ldots, h_P : \mathcal{X} \to \mathcal{C}$ be the base classifiers. The *majority voting classifier*, also called *hard voting classifier* and denoted by $H_v : \mathcal{X} \to \mathcal{C}$, is defined by means of the equation

$$H_v(\mathbf{x}) = \arg\max_{d \in \mathcal{C}} \sum_{\xi=1}^{P} w_\xi \mathcal{I}[h_\xi(\mathbf{x}) = d], \quad \forall \mathbf{x} \in \mathcal{X}, \tag{4.1}$$

where $w_1, \ldots, w_P$ are the weights of the base classifiers and $\mathcal{I}$ is the indicator function, that is,

$$\mathcal{I}[h_\xi(\mathbf{x}) = d] = \begin{cases} 1, & h_\xi(\mathbf{x}) = d, \\ 0, & \text{otherwise}. \end{cases} \tag{4.2}$$

When $\mathcal{C} = \{-1, +1\}$, the majority voting ensemble classifier given by (4.1) can be written alternatively as

$$H_h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{\xi=1}^{P} w_\xi h_\xi(\mathbf{x}) \right), \quad \forall \mathbf{x} \in \mathcal{X}, \tag{4.3}$$

whenever $\sum_{\xi=1}^{P} w_\xi h_\xi(\mathbf{x}) \neq 0$ (Ferreira and Figueiredo, 2012).

4.3 Ensemble Based on Bipolar Associative Memories

Now, let us introduce the ensemble classifiers based on the RCNNs models. In analogy to the majority voting ensemble classifier, the RCNNs-based ensemble classifier is formulated using only the base classifiers $h_1, \ldots, h_P : \mathcal{X} \to \mathbb{B}$. Precisely, consider a training set $\mathcal{T} = \{(\boldsymbol{t}_i, d_i) : i = 1, \ldots, M\} \subset \mathcal{X} \times \mathbb{B}$ and let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\} \subset \mathcal{X}$ be a batch of input samples that we want to classify. We first define the fundamental memories as follows for all $\xi = 1, \ldots, P$:

$$\mathbf{u}^\xi = [h_\xi(\boldsymbol{t}_1), \ldots, h_\xi(\boldsymbol{t}_M), h_\xi(\mathbf{x}_1), \ldots, h_\xi(\mathbf{x}_L)]^T \in \mathbb{B}^{M+L}. \tag{4.4}$$

In words, the ξth fundamental memory is obtained by concatenating the outputs of the ξth base classifier evaluated at the M training samples and the L input samples. Bipolar RCNNs are synthesized using the fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$ and it is initialized at the state vector

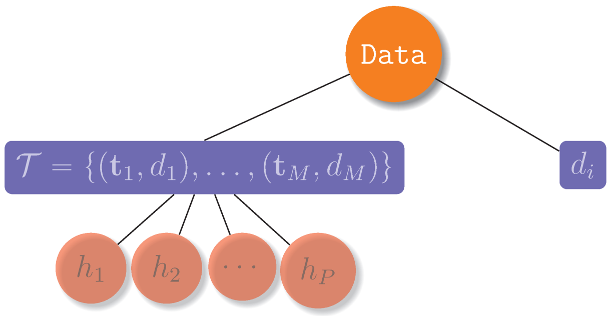
$$\boldsymbol{z}(0) = [d_1, d_2, \ldots, d_M, \underbrace{0, 0, \ldots, 0}_{L-\text{components}}]^T. \tag{4.5}$$

Note that the first M components of the initial state $\boldsymbol{z}(0)$ correspond to the targets in the training set $\mathcal{T}$. The last L components of $\boldsymbol{z}(0)$ are zero, a neutral element different from the class labels. The inital state $\boldsymbol{z}(0)$ is presented as input to the associative memory and the last L components of the recalled vector $\mathbf{y}$ yield the class label of the batch of input samples $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$. In mathematical terms, the RCNN-based ensemble classifier $H_a : \mathcal{X} \to \mathbb{B}$ is defined by means of the equation

$$H_a(\mathbf{x}_i) = y_{M+i}, \quad \forall \mathbf{x}_i \in X, \tag{4.6}$$

where $\mathbf{y} = [y_1, \ldots, y_M, y_{M+1}, \ldots, y_{M+L}]^T$ is the limit of the sequence $\{\boldsymbol{z}(t)\}_{t \geqslant 0}$ given by (1.24) and (1.25). In order to clarify the previous explanation, let us summarize the algorithm of the RCNN-ensemble classifier using a diagram. In Figures 29 and 30 the red color indicates training mode and the gray color a fixed classifier. Given a training set $\mathcal{T} = \{(\boldsymbol{t}_1, d_1), \ldots, (\boldsymbol{t}_M, d_M)\}$ and a batch of inputs $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$ we want to classify, we proceed as follows:

1. First, we train the base classifiers, as it is shown in Figure 29. Also, we keep the desired outputs d_i's to construct the input vector for the RCNN model.



Data
$\mathcal{T} = \{(\mathbf{t}_1, d_1), \ldots, (\mathbf{t}_M, d_M)\}$
d_i
h_1
h_2
$\cdots$
h_P

Figure 29 – Training for the RCNN ensemble method.

2. For each trained base classifier h_ξ, we concatenate the values $h_\xi(\mathbf{t}_1), \ldots, h_\xi(\mathbf{t}_L)$ and $h_\xi(\mathbf{x}_1), \ldots, h_\xi(\mathbf{x}_L)$ obtaining a fundamental memory $\mathbf{u}^\xi$. In Figure 30, green and cyan colors represent the trained base classifiers' outputs over the train data set and the samples correspondingly.

3. Using the vectors $\mathbf{u}^1, \ldots, \mathbf{u}^P$, we construct the fundamental memory matrix U as it is shown in Figure 31. Besides, we construct the input vector for the RCNN model given by

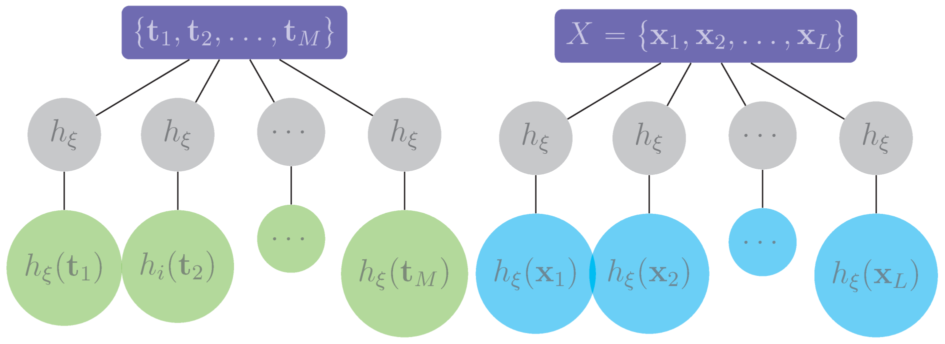


Figure 30 – Storing the output generated by the base classifiers over the training data $\mathcal{T}$ and the input samples X for each h_ξ with $\xi = 1, \ldots, P$.

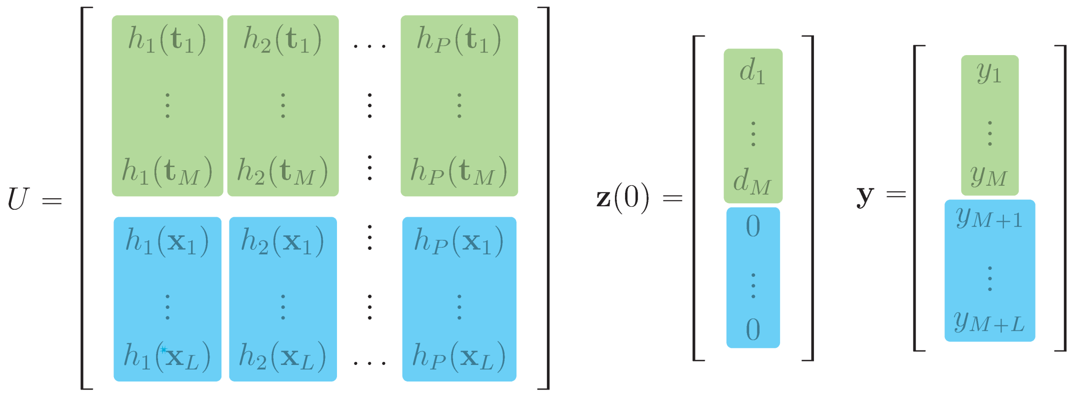

$U =$
$h_1(\mathbf{t}_1)$ $h_2(\mathbf{t}_1)$ $\ldots$ $h_P(\mathbf{t}_1)$
$h_1(\mathbf{t}_M)$ $h_2(\mathbf{t}_M)$ $h_P(\mathbf{t}_M)$
$h_1(\mathbf{x}_1)$ $h_2(\mathbf{x}_1)$ $h_P(\mathbf{x}_1)$
$h_1(\mathbf{x}_L)$ $h_2(\mathbf{x}_L)$ $\ldots$ $h_P(\mathbf{x}_L)$
$\mathbf{z}(0) =$
d_1 d_M 0 0
$\mathbf{y} =$
y_1 y_M y_{M+1} y_{M+L}

Figure 31 – Configuration of the fundamental memory matrix U and the input vector. The colors green and cyan match with the corresponding dimensions between the columns of U and the input vector $\mathbf{z}(0)$ and the output $\mathbf{y}$.

(4.5). We preserve the same colors of Figure 31 to show explicitly the dimension relations to construct U and $z(0)$.

4. The output vector $\mathbf{y}$ contains the classes obtained by using the RCNN model. The last L components $y_{M+1}, \ldots, y_{M+L}$ are the output classes for the $\mathbf{x}_1 \ldots, \mathbf{x}_L$ input examples.

There are two hyperparameters: the exponential parameter α and the maximum iteration number for the RCNN model. The first parameter can be adjusted using an optimization algorithm or set it as a fixed value. We set two different maximum number of iterations in the computational experiments: 1 or 10 iterations. The stopping criterion is based on the number of maximum iterations and the distance between consecutive state vectors $\mathbf{z}(t)$ and $\mathbf{z}(t+1)$ that is impose $||\mathbf{z}(t+1) - \mathbf{z}(t)|| < \tau$ with $\tau = 1.e - 4$.

In the following, we point out the relationship between the bipolar RCNN-based ensemble classifier and the majority voting ensemble described by (4.3). Let $\mathbf{y}$ be the vector recalled by the RCNN fed by the input $z(0)$ given by (4.5), that is, $\mathbf{y}$ is a stationary state of the RCNN. From (1.25), (1.24), and (4.4), the output of the RCNN-based ensemble classifier satisfies

$$H_a(\mathbf{x}_i) = \operatorname{sgn}\left(\sum_{\xi=1}^{P} w_\xi h_\xi(\mathbf{x}_i)\right), \tag{4.7}$$

where

$$w_\xi = f\left(\frac{1}{M+L}\sum_{i=1}^{M+L} y_i u_i^\xi\right), \quad \forall \xi = 1,\ldots,P. \tag{4.8}$$

From (4.7), the bipolar RCNN-based ensemble classifier can be viewed as a weighted majority voting classifier. Furthermore, the weight w_ξ depends on the similarity between the ξth base classifier h_ξ and the ensemble classifier H_a. Precisely, let us define the similarity between two binary classifiers $H, h : \mathcal{X} \to \mathbb{B}$ on a set of samples S by means of the equation

$$\mathtt{Sim}(H,h) = \frac{1}{\text{Card}(S)}\sum_{s\in S}\mathcal{I}\big[h(\boldsymbol{s}) = H(\boldsymbol{s})\big]. \tag{4.9}$$

Using (4.9), we can state the following theorem:

Theorem 4.1. *The weights of the RCNN-based ensemble classifier given by* (4.7) *satisfies the following identities for all $\xi = 1,\ldots,P$:*

$$w_\xi = f\big(1 - 2\cdot\mathtt{Sim}(H_a,h_\xi)\big), \tag{4.10}$$

where the similarity in (4.10) *is evaluated on the union of all training and input samples, that is, on $S = X \cup T = \{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_M\} \cup \{\mathbf{x}_1,\ldots,\mathbf{x}_L\}$.*

Proof. Since we are considering a binary classification problem, the similarity between the ensemble H_a and the base classifier h_ξ on $S = X \cup T$, with $N = \text{Card}(S) = M + L$, satisfies the following identities:

$$\begin{aligned}\mathtt{Sim}(H,h) &= 1 - \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}[h(\boldsymbol{s}_i) \neq H_a(\boldsymbol{s}_i)] = 1 - \frac{1}{4N}\sum_{i=1}^{N}\big(h(\boldsymbol{s}_i) - H_a(\boldsymbol{s}_i)\big)^2\\ &= 1 - \frac{1}{2N}\sum_{i=1}^{N}\big(1 - H_a(\boldsymbol{s}_i)h(\boldsymbol{s}_i)\big) = \frac{1}{2}\left(1 - \frac{1}{N}\sum_{i=1}^{N}H_a(\boldsymbol{s}_i)h(\boldsymbol{s}_i)\right)\end{aligned}$$

Equivalently, we have

$$\frac{1}{\text{Card}(S)}\sum_{s\in S}H(\boldsymbol{s})h(\boldsymbol{s}) = 1 - 2\cdot\mathtt{Sim}(H,h). \tag{4.11}$$

Now, from (2.22), (4.6), and (4.11), we obtain the following identities:

$$w_\xi = f\left(\frac{1}{N}\sum_{i=1}^{N} y_i u_i^\xi\right) = f\big(1 - 2\cdot\mathtt{Sim}(H_a,h_\xi)\big),$$

which concludes the proof. □

Theorem 4.1 shows that the RCNN-based ensemble classifier is a majority voting classifier whose weights depend on the similarity between the base classifiers and the ensemble itself. In fact, in view of the dynamic nature of the RCNN model, H_a is obtained by a recurrent consult and vote scheme. Moreover, at the first step, the weights depend on the accuracy of the base classifiers.

4.3.1 Computational Experiments

In this section, we perform some computational experiments to evaluate the performance of the proposed RCNN-based ensemble classifiers for binary classification tasks. Precisely, we considered the RCNN-based ensembles obtained using the identity and the exponential as the activation function f. The parameter α of the exponential activation function has been either set to $\alpha = 1$ or it has been determined using a grid search on the set $\{10^{-2}, 10^{-1}, 0.5, 1, 5, 10, 20, 50\}$ with 10-fold cross-validation on the training set. The RCNN-based ensemble classifiers have been compared with AdaBoost, gradient boosting, and random forest ensemble classifiers, all available at the python's `scikit-learn API` (`sklearn`) (Pedregosa et al., 2011).

First of all, we trained AdaBoost and gradient boosting ensemble classifiers using the default parameters of `sklearn`. Recall that boosting ensemble classifiers are developed incrementally by adding base classifiers to reduce the number of misclassified samples (Kuncheva, 2014). Also, we trained the random forest classifier with 30 base classifiers ($P = 30$) (Breiman, 2001). Recall that the base classifiers of the random forest are decision trees obtained using bagging and random subspace techniques (Breiman, 1996; Ho, 1998). Then, we used the base classifiers from the trained random forest ensemble to define the RCNN-based ensemble. In other words, the same base classifiers $h_1, \ldots, h_{30}$ are used in the random forest and the RCNN-based classifiers. The difference between the two ensemble classifiers resides in the combining rule. Recall that the random forest combines the base classifiers using majority voting. From the computational point of view, training the random forest and the RCNN-ensemble classifiers required similar resources. Moreover, despite the consult and vote scheme of the RCNN-based ensemble, they have not been significantly more expensive than the random forest classifier. The grid search used to fine-tune the parameter α of the exponential RCNN-based ensemble is the major computational burden in this computational experiment.

For the comparison of the ensemble classifiers, we considered 28 binary classification problems from the OpenML repository (Vanschoren et al., 2013). These binary classification problems can be obtained using the command `fetch_openml` from `sklearn`. We would like to point out that missing data has been handled before splitting the data set into training and test sets using the command `SimpleImputer` from `sklearn`. Also, we pre-processed the data using the `StandardScaler` transform. Therefore, each feature is normalized by subtracting the mean and dividing by the standard deviation, both computed using only the training set. Furthermore, since some data sets are unbalanced, we used the F-measure to evaluate quantitatively the performance of a certain classifier. Table 4 shows the mean and the standard deviation of the F-measure obtained from the ensemble classifiers using stratified 10-fold cross-validation. The largest F-measures for each data set have been typed using boldface. Note that the exponential RCNN-based ensemble classifier with grid search produced the largest F-measures in 11 of the 28 data sets. In particular, the exponential RCNN with grid search produced outstanding F-measures on the "Monks-2" and "Egg-Eye-State" data sets. For a better

Table 4 – Mean and standard deviation of the F-measures produced by ensemble classifiers using stratified 10-fold cross-validation.

Data set	**AdaBoost**	**Gradient Boosting**	**Random Forest**	**Identity RCNN**	**Exponential RCNN**	**Exp. RCNN + Grid Search**
Arsene	84.0 ± 5.9	$\mathbf{86.2 \pm 7.6}$	81.5 ± 8.9	83.8 ± 8.4	83.8 ± 8.4	85.2 ± 10.2
Australian	82.1 ± 3.4	$\mathbf{85.8 \pm 3.8}$	85.4 ± 3.4	85.3 ± 2.9	85.3 ± 2.9	85.0 ± 2.9
Banana	67.9 ± 2.1	88.1 ± 1.6	88.0 ± 1.3	$\mathbf{88.2 \pm 1.2}$	88.2 ± 1.2	87.2 ± 1.2
Banknote	$\mathbf{99.6 \pm 0.4}$	99.5 ± 0.9	99.3 ± 0.7	99.2 ± 0.7	99.2 ± 0.7	98.9 ± 0.9
Blood Transfusion	$\mathbf{43.0 \pm 13.1}$	37.9 ± 11.2	32.3 ± 10.4	33.3 ± 10.6	33.3 ± 10.6	32.5 ± 8.2
Breast Cancer Wisconsin	94.7 ± 2.0	95.2 ± 2.4	94.9 ± 3.4	$\mathbf{95.4 \pm 2.9}$	95.1 ± 3.3	95.2 ± 4.2
Chess	96.5 ± 1.1	97.9 ± 0.8	99.0 ± 0.5	99.0 ± 0.6	99.0 ± 0.6	$\mathbf{99.2 \pm 0.4}$
Colic	87.1 ± 6.4	86.7 ± 7.4	88.7 ± 5.7	88.6 ± 5.4	88.6 ± 5.4	$\mathbf{88.9 \pm 4.6}$
Credit Approval	86.4 ± 2.9	86.9 ± 3.2	88.4 ± 2.8	$\mathbf{88.4 \pm 2.5}$	$\mathbf{88.4 \pm 2.5}$	88.3 ± 2.3
Credit-g	82.3 ± 2.5	84.2 ± 2.8	83.7 ± 2.4	$\mathbf{84.3 \pm 2.2}$	$\mathbf{84.3 \pm 2.2}$	83.9 ± 1.8
Cylinder Bands	78.3 ± 4.8	84.0 ± 4.8	83.0 ± 6.6	83.3 ± 6.4	83.3 ± 6.4	$\mathbf{87.0 \pm 4.2}$
Diabetes	63.1 ± 5.2	65.1 ± 6.5	63.9 ± 8.8	$\mathbf{65.6 \pm 8.2}$	$\mathbf{65.6 \pm 8.2}$	62.4 ± 7.8
Egg-Eye-State	70.1 ± 1.3	78.0 ± 0.9	91.5 ± 0.7	91.8 ± 0.8	91.8 ± 0.8	$\mathbf{92.9 \pm 0.8}$
Haberman	$\mathbf{35.4 \pm 9.5}$	30.8 ± 14.2	27.4 ± 13.4	30.6 ± 9.6	30.6 ± 9.6	34.9 ± 12.9
Hill-Valley	40.9 ± 5.4	52.9 ± 7.3	54.9 ± 4.6	56.6 ± 3.8	56.6 ± 4.0	$\mathbf{59.1 \pm 6.2}$
Internet Advertisements	98.0 ± 0.3	98.6 ± 0.3	$\mathbf{98.8 \pm 0.4}$	98.7 ± 0.4	98.7 ± 0.4	98.7 ± 0.5
Ionosphere	94.3 ± 1.7	94.4 ± 2.0	94.2 ± 2.5	94.0 ± 2.5	94.0 ± 2.5	$\mathbf{94.7 \pm 2.7}$
MOFN-3-7-10	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.8 ± 0.2	99.7 ± 0.3	99.7 ± 0.3	99.7 ± 0.5
Monks-2	0.0 ± 0.0	69.3 ± 8.7	93.1 ± 3.3	93.5 ± 3.3	93.5 ± 3.3	$\mathbf{98.5 \pm 2.7}$
Phoneme	68.3 ± 3.0	75.4 ± 2.4	84.0 ± 3.0	84.1 ± 2.7	84.1 ± 2.7	$\mathbf{85.7 \pm 2.0}$
Pishing Websites	94.4 ± 0.4	95.3 ± 0.5	97.5 ± 0.6	97.4 ± 0.6	97.4 ± 0.6	$\mathbf{97.5 \pm 0.5}$
Sick	78.3 ± 6.4	88.8 ± 3.9	87.5 ± 3.1	88.6 ± 3.9	88.6 ± 3.9	$\mathbf{89.7 \pm 3.6}$
Sonar	$\mathbf{83.9 \pm 8.0}$	81.3 ± 6.2	81.9 ± 11.4	83.3 ± 11.1	83.3 ± 11.1	83.2 ± 11.1
Spambase	91.8 ± 1.5	93.1 ± 1.7	$\mathbf{94.2 \pm 1.1}$	94.0 ± 1.2	94.1 ± 1.2	94.0 ± 1.2
Steel Plates Fault	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.0 ± 0.8	99.2 ± 0.6	99.2 ± 0.6	99.4 ± 0.7
Tic-Tac-Toe	84.5 ± 2.6	94.8 ± 2.1	95.6 ± 1.2	95.5 ± 1.2	95.5 ± 1.2	$\mathbf{96.5 \pm 1.5}$
Titanic	$\mathbf{58.8 \pm 4.3}$	53.8 ± 4.4	53.6 ± 4.2	53.6 ± 4.2	53.6 ± 4.2	53.8 ± 4.4
ilpd	$\mathbf{41.4 \pm 11.4}$	35.3 ± 15.1	35.1 ± 15.8	37.5 ± 16.6	37.5 ± 16.6	33.5 ± 14.6

comparison of the ensemble classifiers, we followed Demšar's recommendations to compare multiple classifier models using multiple data sets (Demšar, 2006). The Friedman test rejected the hypothesis that there is no difference between the ensemble classifiers. A visual interpretation of the outcome of this computational experiment is provided in Figure 32 with the Hasse diagram of the non-parametric Wilcoxon signed-rank test with a confidence level at 95% (Burda, 2013; Weise and Chiong, 2015). In this diagram, an edge means that the classifier on the top statistically outperformed the classifier on the bottom. The outcome of this analysis confirms that the RCNN-based ensemble classifiers statistically outperformed the other ensemble methods: AdaBoost, gradient boosting, and random forest.

As to the computational effort, Figure 33 shows the average time required by the ensemble classifiers for the prediction of a batch of testing samples. Note that the most expensive method is the identity RCNN-based ensemble classifier while the gradient boosting is the cheapest. The exponential RCNN-based ensemble is less expensive than the AdaBoost and quite comparable to the random forest classifier.

Finally, note from Table 4 that some problems such as the "Banknote" and the "MOFN-3-7-10" data sets are quite easy while others such as the "Haberman" and "Hill Valey"

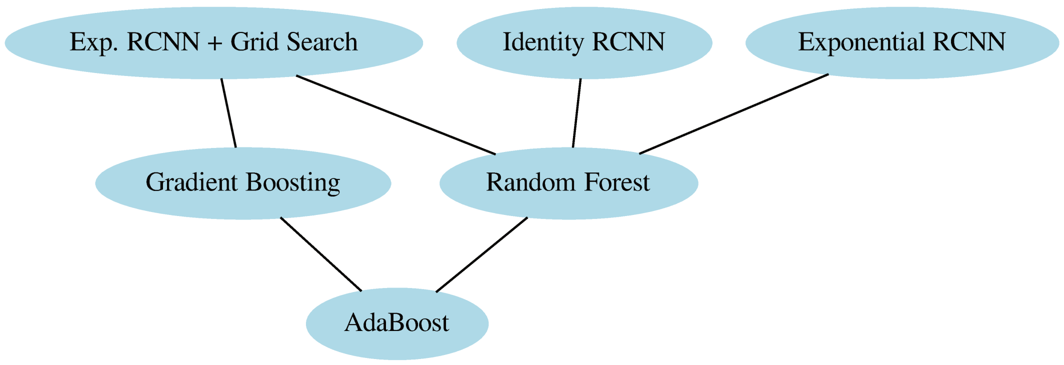


Figure 32 – Hasse diagram of Wilcoxon signed-rank test with a confidence level at 95%.

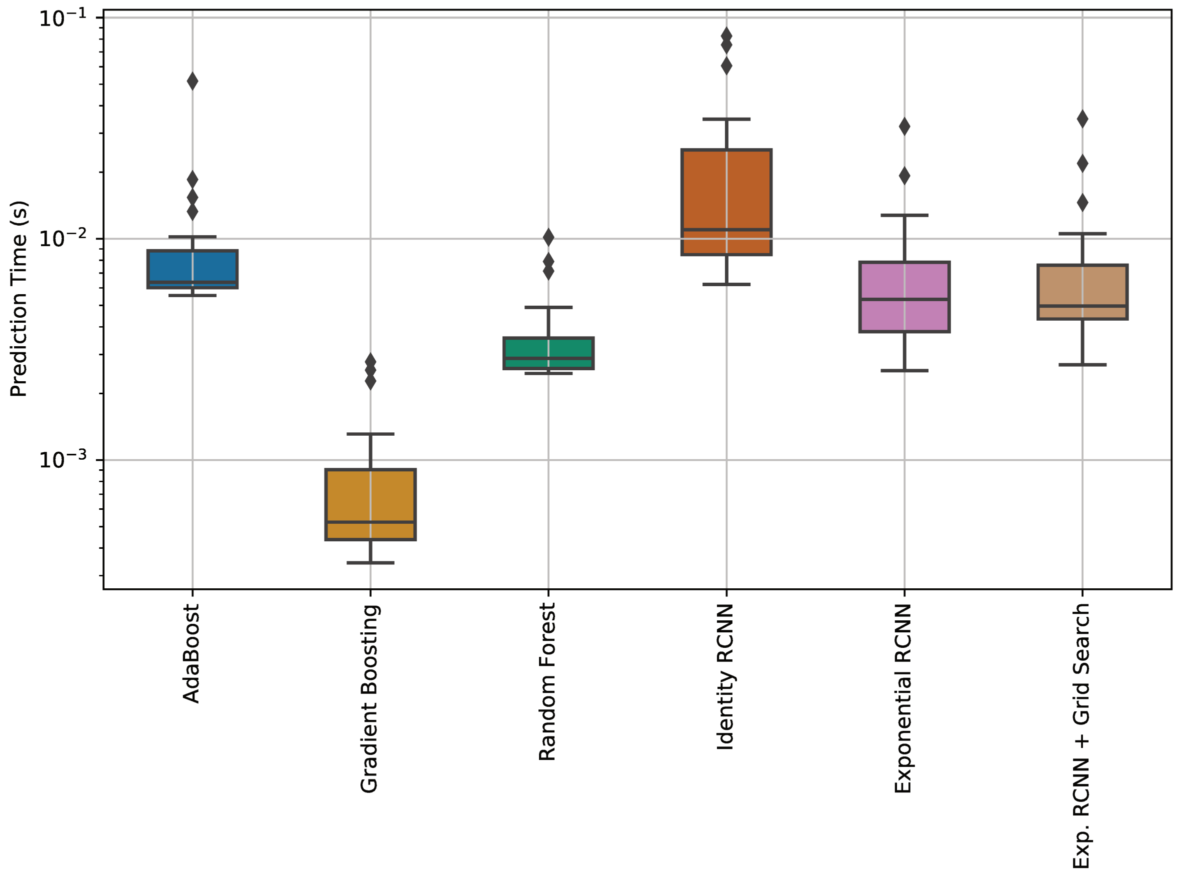


Figure 33 – Box-plot of the average time for prediction of batch of input samples.

are very hard. In order to circumvent the difficulties imposed by each data set, Figure 34 shows a box-plot with the normalized F-measure values provided in Table 4. Precisely, for each data set (i.e., each row in Table 4), we subtracted the mean and divided by the standard deviation of the score values. The box-plot in Figure 34 confirms the good performance of the RCNN-based ensemble classifiers, including the exponential RCNN-based ensemble classifier with a grid

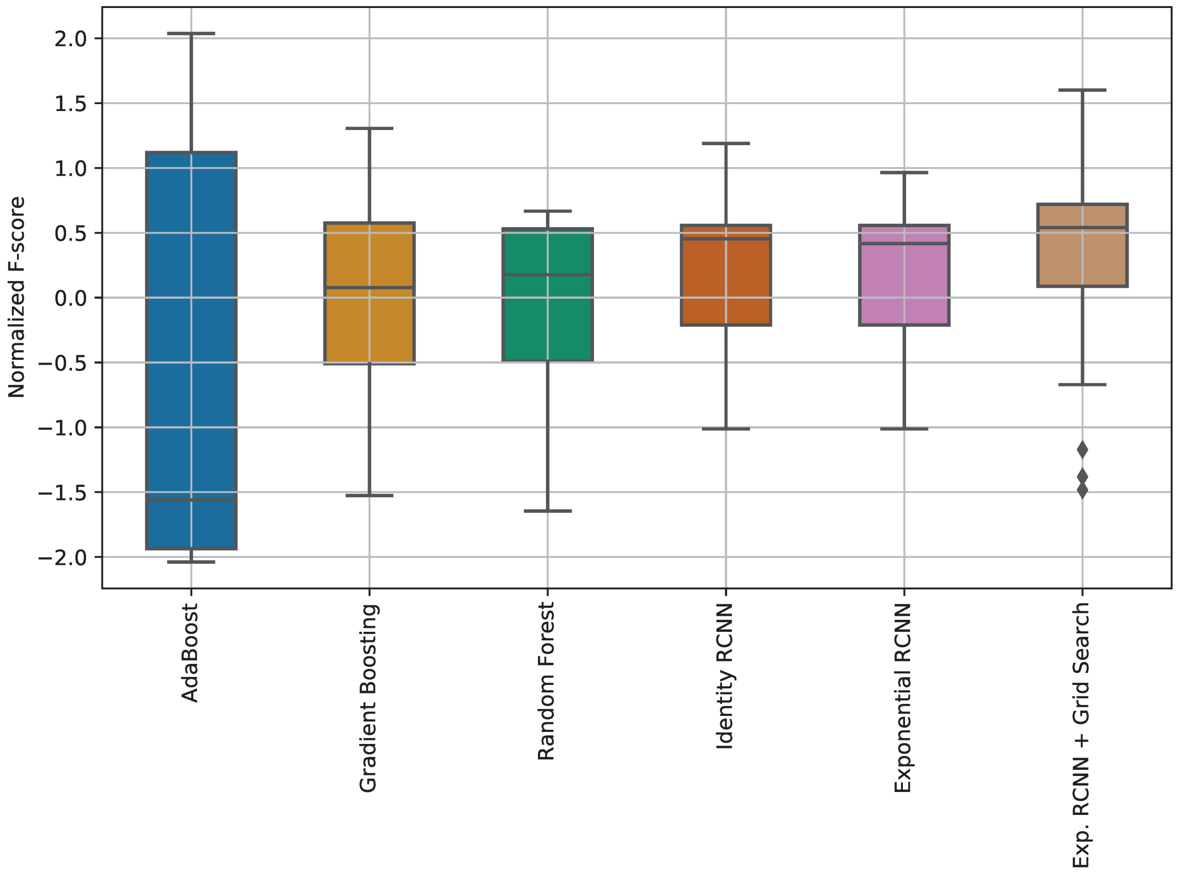


Figure 34 – Box-plot of the normalized F-measures produced by the ensemble classifiers.

search. Concluding, the boxplots shown on Figures 33 and 34 supports the potential application of the RCNN models as an ensemble of classifiers for binary classification problems.

4.4 Multi-Class Ensemble Classifier

Let us extend the previous model in order to obtain a multi-class ensemble classifier. Analogously to the RCNN-based ensemble, we study the complex-valued and quaternion-valued extensions. We use the *complex-valued multi-state* ECNN, and the *quaternion-valued multi-state* ECNN to extend the binary model.

Recall from Chapter 3 that the complex-valued and quaternion-valued multi-state models yielded the best results in pattern reconstruction tasks. Despite not having a trivial relation between the performance of AMs in pattern reconstruction context and the proposed meta-learning voting step, we perform computational experiments to compare the multi-state ECNN-based ensemble with two literature models: gradient boost and random forest classifiers.

On the one hand, for the complex case, we use the encoding given in Section 3.5.3 to transform the predicted classes to complex numbers, where the resolution factor K is equal to the number of classes. Then, we implement the complex-valued multi-state ECNN using the

$\texttt{csgn}$ activation function. On the other hand, for the quaternionic case, we encode the classes $d \in \mathcal{C}$, where $\mathcal{C} = \{1, 2, \dots, n_c\}$, as follows:

$$K = \left\lceil \sqrt{n_c} \right\rceil. \tag{4.12}$$

Then, we compute the remainder and quotient such that

$$d_i = \texttt{quotient}_i \cdot K + \texttt{remainder}_i. \tag{4.13}$$

Note that (4.13) transform the class index into two different integers with a unique representation. The two integers, $\texttt{quotient}_i$ and $\texttt{remainder}_i$, are transformed to complex numbers by using the equations

$$z_{i1} = e^{2\pi(\texttt{remainder}_i)\mathbf{i}/K}, \tag{4.14}$$

and

$$z_{i2} = e^{2\pi(\texttt{quotient}_i)\mathbf{i}/K}. \tag{4.15}$$

Finally, taking the real and complex parts separately, we construct the quaternionic representation $q = [\text{Re}\{z_{i1}\}, \texttt{Img}(z_{i1}), \text{Re}\{z_{i2}\}, \texttt{Img}(z_{i2})]$ that yields a quaternion in the set $\mathcal{S}_t$ given by (3.34). Due to the nature of the transformation, note that we could have more class representations than the number of classes for a given problem. For example, if we have 13 classes, then we have $K = 4$. The possible combinations of pairs $(\texttt{remainder}, \texttt{quotient})$ are 16, obtaining three new classes without representation. In the computational experiments section, we explain how to deal with this problem.

Example 2. *As an example for the quaternion-valued case, suppose that we have 9 classes. Using* (4.13) *we obtain $K = 3$. Also, suppose that we have in the training step the following real-valued fundamental memory matrix for three base classifiers:*

$$U^r = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}. \tag{4.16}$$

Using (4.13), (4.14) *and* (4.15) *we obtain the matrices of integers given by*

$$R = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad Q = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \end{pmatrix}. \tag{4.17}$$

where R and Q are the matrices obtained from the remainder and quotients of the matrix U module K. Then, we apply in an element-wise manner the complex transformation of section 3.5.3 obtaining

$$Z_1 = \begin{pmatrix} -0.5 + 0.866\mathbf{i} & -0.5 + 0.866\mathbf{i} & -0.5 + 0.866\mathbf{i} \\ -0.5 - 0.866\mathbf{i} & -0.5 - 0.866\mathbf{i} & -0.5 - 0.866\mathbf{i} \\ 1. + 0.000\mathbf{i} & 1. + 0.000\mathbf{i} & 1. + 0.000\mathbf{i} \end{pmatrix}, \tag{4.18}$$

and

$$Z_2 = \begin{pmatrix} 1 + 0.000\mathbf{i} & -0.5 + 0.866\mathbf{i} & -0.5 - 0.866\mathbf{i} \\ 1 + 0.000\mathbf{i} & -0.5 + 0.866\mathbf{i} & -0.5 - 0.866\mathbf{i} \\ -0.5 + 0.866\mathbf{i} & -0.5 - 0.866\mathbf{i} & 1. - 0.000\mathbf{i} \end{pmatrix}. \tag{4.19}$$

Finally, we separate the real an imaginary parts of every component to create a quaternionic matrix representation. In this example, we have the 3×3 *quaternionic matrix*

$$U^q = \begin{pmatrix} -0.5 - 0.866\mathbf{i} + 1\mathbf{j} & -0.5 - 0.866\mathbf{i} - 0.5\mathbf{j} + 0.866\mathbf{k} & -0.5 + 0.866\mathbf{i} - 0.5\mathbf{j} - 0.866\mathbf{k} \\ -0.5 - 0.866\mathbf{i} + 1\mathbf{j} & -0.5 - 0.866\mathbf{i} - 0.5\mathbf{j} + 0.866\mathbf{k} & -0.5 - 0.866\mathbf{i} - 0.5\mathbf{j} - 0.866\mathbf{k} \\ 1 - 0.5\mathbf{j} + 0.866\mathbf{k} & 1 - 0.5\mathbf{j} - 0.866\mathbf{k} & 1 + 1\mathbf{j} \end{pmatrix} \tag{4.20}$$

4.4.1 Computational Experiments

We implement the complex-valued multi-state ECNN and quaternion-valued multi-state ECNN to compare the classification capability against the gradient boost and random forest ensemble classifiers, both available at the python's `scikit-learn API` (`sklearn`) (Pedregosa et al., 2011). Analogously, the parameter α has been determined using a grid search on the set $\{10^{-2}, 10^{-1}, 0.5, 1, 5, 10, 20, 50\}$ with 10-fold cross-validation on the training set. Besides the fine-tuned α, we also used $\alpha = 1$ or $\alpha = 10$.

We also used the random forest trained base classifiers to construct the HRCNN-based ensemble classifier. We trained the random forest classifier with 30 base classifiers. For the comparison of the multi-class ensemble classifiers, we use 24 data sets for multi-class classification problems from the OpenML repository (Vanschoren et al., 2013). One essential difference between the encoding in the binary case and in the multi-class case comes from the representation space. In particular, for the quaternion-valued model, we need to handle in a different manner the label encoding of categorical classes, due to the unseen classes that could appear. In that case we obtain an unseen class, we transform the unseen value to a new one, that we recognize as an unknown class. In our implementations, we used the label -1 to represent the unknow class. When we evaluate the model, we do not consider the unknown class as a correct classification. The complex-valued model does not suffer from this issue.

Table 5 shows the mean and the standard deviation of the kappa index obtained from the ensemble classifiers using stratified 10-fold cross-validation. Analogously to the binary RCNN ensemble-based model, missing data has been handled before splitting the data set into training and test sets using the command `SimpleImputer` from `sklearn`. Also, we pre-processed the data using the `StandardScaler` transform. The largest kappa index for each data set have been typed using boldface. Note that the gradient boost and the quaternion-valued multi-state ECNN (QvECNN) produce the largest kappa index. Precisely, the gradient boost method obtained 12 superior kappa index, some shared with the QvECNN with grid search. The QvECNN obtained 11 superior results, sharing some numbers with the complex-valued ECNN,

Table 5 – Mean and standard deviation of the Kappa Index produced by multi-class ensemble classifiers using stratified 10-fold cross-validation.

Data set	**Gradient Boost**	**Random Forest**	**CvHNN**	**CvECNN(1)**	**CvECNN**(10)	**CvECNN + Grid Search**	**QvECNN(1)**	**QvECNN(10)**	**QvECNN + Grid Search**
Wine	**0.975 ± 0.04**	**0.975 ± 0.04**	**0.975 ± 0.04**	**0.975 ± 0.04**	**0.975 ± 0.04**	**0.975 ± 0.04**	**0.975 ± 0.04**	0.967 ± 0.043	**0.975 ± 0.04**
iris	0.93 ± 0.067	0.93 ± 0.067	0.93 ± 0.067	**0.94 ± 0.07**	**0.94 ± 0.07**	**0.94 ± 0.07**	0.93 ± 0.067	0.93 ± 0.082	0.93 ± 0.082
DNA	0.913 ± 0.013	0.913 ± 0.013	0.91 ± 0.014	0.91 ± 0.013	0.894 ± 0.015	0.911 ± 0.012	0.894 ± 0.015	**0.921 ± 0.017**	0.904 ± 0.02
Steel Plates Fault	**0.706 ± 0.037**	0.706 ± 0.037	0.623 ± 0.036	0.624 ± 0.036	0.693 ± 0.029	0.622 ± 0.035	0.692 ± 0.028	0.647 ± 0.034	0.697 ± 0.034
Segment	0.967 ± 0.008	0.967 ± 0.008	0.945 ± 0.013	0.945 ± 0.013	0.964 ± 0.01	0.945 ± 0.013	0.964 ± 0.01	0.951 ± 0.015	**0.969 ± 0.012**
Synthetic control	0.984 ± 0.031	0.984 ± 0.031	0.952 ± 0.046	0.954 ± 0.045	0.982 ± 0.03	0.956 ± 0.044	0.982 ± 0.03	0.964 ± 0.034	**0.988 ± 0.019**
Cardiotocography	1.0 ± 0.0	1.0 ± 0.0	0.997 ± 0.004	0.997 ± 0.004	1.0 ± 0.0	0.997 ± 0.004	1.0 ± 0.0	0.999 ± 0.002	1.0 ± 0.0
Balance Scale	0.695 ± 0.068	0.695 ± 0.068	0.698 ± 0.067	0.698 ± 0.067	0.694 ± 0.087	0.693 ± 0.068	0.693 ± 0.088	0.667 ± 0.055	0.659 ± 0.058
Satimage	0.892 ± 0.01	0.892 ± 0.01	0.837 ± 0.011	0.837 ± 0.012	0.885 ± 0.012	0.837 ± 0.012	0.885 ± 0.012	0.851 ± 0.007	**0.893 ± 0.008**
CNAE-9	0.901 ± 0.033	0.901 ± 0.033	0.83 ± 0.038	0.83 ± 0.038	0.895 ± 0.033	0.831 ± 0.04	0.897 ± 0.032	0.871 ± 0.047	**0.914 ± 0.029**
Arrhythmia	0.522 ± 0.108	0.522 ± 0.108	0.383 ± 0.106	0.372 ± 0.105	0.482 ± 0.096	0.41 ± 0.107	0.489 ± 0.108	0.418 ± 0.095	**0.528 ± 0.083**
CJS	**0.724 ± 0.029**	0.724 ± 0.029	0.678 ± 0.024	0.677 ± 0.026	0.707 ± 0.034	0.681 ± 0.027	0.707 ± 0.032	0.692 ± 0.026	0.722 ± 0.028
Gesture	0.538 ± 0.018	0.538 ± 0.018	0.457 ± 0.022	0.457 ± 0.023	0.511 ± 0.018	0.456 ± 0.022	0.511 ± 0.018	0.503 ± 0.015	**0.547 ± 0.025**
Anauthor	**0.984 ± 0.01**	**0.984 ± 0.01**	0.979 ± 0.014	0.981 ± 0.013	0.984 ± 0.013	0.977 ± 0.016	0.983 ± 0.012	0.977 ± 0.014	0.979 ± 0.016
Morphologic	**0.664 ± 0.018**	0.664 ± 0.018	0.538 ± 0.035	0.542 ± 0.033	0.643 ± 0.021	0.541 ± 0.038	0.643 ± 0.021	0.581 ± 0.027	0.645 ± 0.028
Optdigits	**0.977 ± 0.007**	**0.977 ± 0.007**	0.916 ± 0.008	0.916 ± 0.008	0.956 ± 0.007	0.917 ± 0.008	0.956 ± 0.007	0.932 ± 0.012	0.966 ± 0.009
Micro-Mass	**0.858 ± 0.049**	**0.858 ± 0.049**	0.583 ± 0.083	0.583 ± 0.084	0.755 ± 0.045	0.581 ± 0.075	0.753 ± 0.03	0.643 ± 0.07	0.819 ± 0.038
Texture	**0.802 ± 0.033**	**0.802 ± 0.033**	0.079 ± 0.024	0.083 ± 0.021	0.393 ± 0.029	0.087 ± 0.022	0.397 ± 0.019	0.128 ± 0.026	0.504 ± 0.04
Miceprotein	**0.996 ± 0.005**	**0.996 ± 0.005**	0.952 ± 0.029	0.952 ± 0.029	0.978 ± 0.015	0.957 ± 0.027	0.98 ± 0.015	0.968 ± 0.007	**0.99 ± 0.006**
Chess	0.62 ± 0.007	0.62 ± 0.007	0.622 ± 0.008	0.621 ± 0.008	0.621 ± 0.009	**0.622 ± 0.008**	0.621 ± 0.009	0.593 ± 0.008	0.591 ± 0.008
Gas	**0.993 ± 0.002**	**0.993 ± 0.002**	0.989 ± 0.002	0.99 ± 0.002	0.992 ± 0.002	0.99 ± 0.002	0.992 ± 0.002	**0.992 ± 0.003**	**0.993 ± 0.002**
Wave	**0.772 ± 0.012**	**0.772 ± 0.012**	0.768 ± 0.014	0.769 ± 0.013	0.763 ± 0.011	0.769 ± 0.013	0.762 ± 0.011	0.771 ± 0.019	0.763 ± 0.018
OVA-Breast	0.91 ± 0.023	0.91 ± 0.023	0.906 ± 0.022	0.906 ± 0.022	0.906 ± 0.022	0.906 ± 0.022	0.906 ± 0.022	0.9 ± 0.029	0.9 ± 0.029
Indian pines	0.867 ± 0.015	0.867 ± 0.015	0.852 ± 0.017	0.853 ± 0.018	0.862 ± 0.016	0.853 ± 0.018	0.862 ± 0.016	0.87 ± 0.015	**0.877 ± 0.015**

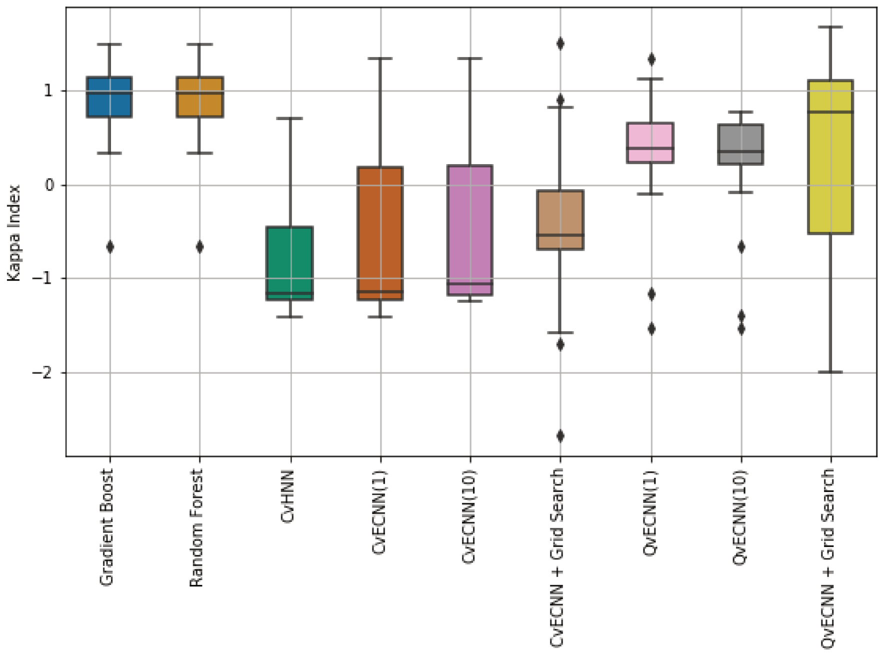


Figure 35 – Normalized box-plot of the kappa index for the multi-class ensemble classifiers.

gradient boost, and random forest. We can observe that gradient boost and the QvECNN obtained the same performance in the `Wine` and `Gas` data sets. Nevertheless, the literature models obtain an outstanding performance in the `texture` data set.

Concluding, the complex and quaternion-valued models do not achieved outstanding performance; they did not statistically outperformed the literature methods. However, the associative memory models were not created with that purpose, yet they are still competitive in multiclass classification tasks. A visual interpretation of the outcome of this computational experiment is provided in Figure 36 with the Hasse diagram of the non-parametric Wilcoxon

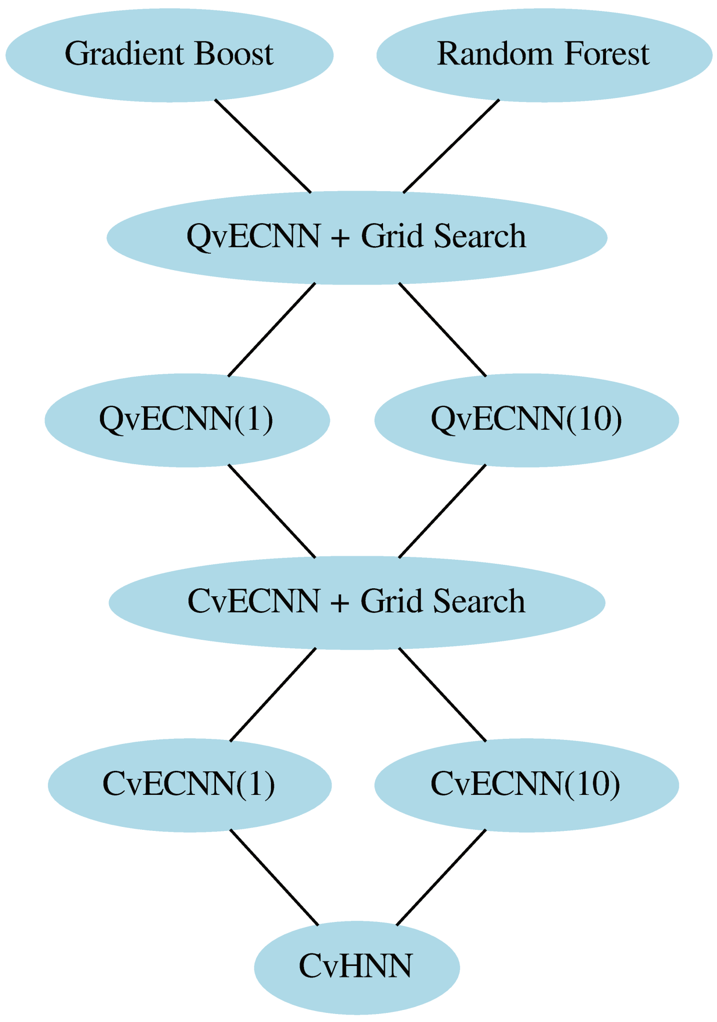


Figure 36 – Hasse diagram of Wilcoxon signed-rank test with a confidence level at 95% for the multi-class ensemble classifiers.

signed-rank test with a confidence level at 95%, and Figure 36 with the normalized box-plot of the kappa index. In the diagram of Figure 36, an edge means that the classifier on the top statistically outperformed the classifier on the bottom. The outcome of this analysis confirms that the RCNN-based ensemble classifiers, using complex and quaternionic algebras, statistically do not outperformed the gradient boosting and the random forest. Finally, we observe that the relation between classes and the resolution factor does not has evident correlation with the performance of the HRCNN-based ensemble. The box-plot in Figure 35 confirms the aforementioned performance of RCNN-based ensembles, the model with the best mean kappa index is the QvECNN+`GridSearch`, and the CvHNN has lower performance. In particular, the grid search QvECNN has more variance than the literature model and the best mean kappa index among the other HRCNN-based ensemble.

5 Conclusions

In this thesis, we presented two new hypercomplex-valued recurrent neural networks: the hypercomplex recurrent correlation neural networks (HRCNNs) and the hypercomplex recurrent projection neural networks (HRPNNs). First, we extend the RCNN model to a broad class of hypercomplex numbers. Hypercomplex-valued neural networks constitute an emerging research area. Besides their natural capability to treat high-dimensional data as single entities, some hypercomplex-valued neural networks can cope with phase or rotation information (Aizenberg, 2011; Hirose, 2012; Parcollet et al., 2019) while others exhibit fast learning rules (Nitta and Kuroe, 2018). In particular, RCNNs have been introduced by Chiueh and Goodman (1991) as an improved version of the famous correlation-based Hopfield neural network. In fact, some RCNNs can reach the storage capacity of an ideal associative memory (Hassoun and Watta, 1996). Furthermore, they are closely related to the dense associative memories of Krotov and Hopfield (2016) as well as the support vector machines and the kernel trick (García and Moreno, 2004a,b; Perfetti and Ricci, 2008). In particular, the exponential correlation neural network (ECNN), besides been amiable for very-large-scale integration (VLSI) implementation, has a Bayesian interpretation (Chiueh and Goodman, 1993; Hancock and Pelillo, 1998). Second, HRPNNs are obtained by combining the projection rule with the HRCNNs. This models generalizes the bipolar Hopfield neural networks by projection rule when the excitation function f is the identity function (see Theorem 2.3 from section 2). In contrast to the HRCNNs, however, HRPNNs always exhibit optimal storage capacity (see Theorem 2.1 from section 2). Moreover, bipolar RPNN and the recurrent kernel associative memory (RKAM) models coincide under mild conditions detailed in Theorem 2.4 from Section 2. The computational experiments provided in Sections 2.3.2 and 3.5 show that the storage capacity and noise tolerance of HRPNNs (including real-valued case) are greater than or equal to the storage capacity and noise tolerance of their corresponding HRCNNs.

Precisely, in this work, we reviewed the basic concepts on hypercomplex-number systems. Briefly, the key notions for the stability analysis of hypercomplex-valued RCNNs is the class of $\mathcal{B}$-functions (see Definition 2) (de Castro and Valle, 2020). Theorem 3.1 from section 3.3 shows that hypercomplex-valued RCNNs always settle at an equilibrium, using either synchronous or asynchronous update modes, if the activation function in the output layer is a $\mathcal{B}$-function. Examples of the bipolar, complex, hyperbolic, quaternion, and octonion-valued RCNNs are given in Section 3.5 to illustrate the theoretical results. Furthermore, Section 3.5 presents a detailed experiment concerning the storage and recall of natural gray-scale images. The quaternion-valued ECNN with the twin-multistate activation function exhibited the largest noise tolerance in this experiment. Also, the asynchronous update yielded a probability of successful recall larger than the synchronous update. Analogously, we implement some computational experiments to explore the HRPNN models to a broad class of hypercomplex numbers. We compare the new models with their corresponding HRCNNs. In particular, we implement hypercomplex-valued EPNNs to compare the noise tolerance in relation to the hypercomplex-valued ECNNs. Also, we implement HRPNN models as associative memories. In the hypercomplex-valued EPNN case we do not know an energy function to analyse the dynamics of the model. However, we repeat the same examples given in section 3.5 to analyze the dynamical behavior and storage capacity of the HEPNNs. We observe that the HRPNNs usually have a better noise tolerance than their corresponding HRCNNs. In particular, the quaternion-valued EPNN with the twin-multistate activation function exhibited the largest noise tolerance in the computational experiments.

Finally, we provide a bridge between ensemble methods and associative memories. In general terms, an ensemble method reduces variance and improve the accuracy and robustness by combining a group of base predictors (Zhang and Ma, 2012; Kuncheva, 2014). The rule used to combine the base predictors is one important issue in the design of an ensemble method. We proposed combining the base predictors using an associative memory. Associative memory is a model designed for the storage and recall of a set of vectors (Hassoun and Watta, 1997). Furthermore, an associative memory should be able to retrieve a stored item from a corrupted or partial version of it. In an ensemble method, the memory model is designed for the storage of evaluations of the base classifiers. The associative memory is then fed by a vector with the target of training data as well as the unknown predictions. The output of the ensemble method is obtained from the vector retrieved by the memory. Specifically, we presented ensemble methods based on the recurrent correlation associative memories for classifications. Theorem 4.1 shows that the RCNN-based ensemble model yields a majority voting classifier whose weights are obtained by a recurrent consult and vote scheme. Moreover, the weights depend on the similarity between the base classifiers and the resulting ensemble. Computational experiments using decision tree as the base classifiers revealed an outstanding performance of the exponential RCNN-based ensemble classifier combined with a grid search strategy to fine-tune its parameter. The exponential RCNN-based ensemble, in particular, outperformed the traditional AdaBoost, gradient boosting, and random forest classifiers for binary classification tasks. Also, we studied the multi-class extension of RCNN-based ensemble models, by using complex and quaternions, observing a comparable but lower performance than gradient boost and random forest classifiers.

In the future, we plan to study hypercomplex-valued RCNN defined on different hypercomplex-number systems as well as models with alternative activation functions. Moreover, we plan to deepen the study of the dynamics of the HRPNNs. In relation to the classification application, we plan to study other possibilities to construct a space representation for multi-classes and deepen in the issues related to this meta-algorithm.

Bibliography

I. Aizenberg, C. Moraga, and D. Paliy. A Feedforward Neural Network based on Multi-Valued Neurons. In B. Reusch, editor, *Computational Intelligence, Theory and Applications*, volume 33 of *Advances in Soft Computing*, pages 599–612. Springer Berlin Heidelberg, 2005. Cited in page 32.

I. N. Aizenberg. *Complex-Valued Neural Networks with Multi-Valued Neurons*, volume 353 of *Studies in Computational Intelligence*. Springer, 2011. doi: 10.1007/978-3-642-20353-4. Cited in page 97.

N. N. Aizenberg and I. N. Aizenberg. CNN based on multivalued neuron as a model of associative memory for gray-scale images. In *Proceedings of the 2nd International Workshop on Cellular Neural Networks and Their Applications*, pages 36–42, 1992. Cited 3 times in page 20, 29, and 57.

M. Arbib. *Brains, Machines, and Mathematics*. Springer-Verlag, New York, NY, 1987. Cited in page 18.

P. Arena, L. Fortuna, G. Muscato, and M. Xibilia. Multilayer Perceptrons to Approximate Quaternion Valued Functions. *Neural Networks*, 10(2):335–342, 1997. doi: 10.1016/S0893-6080(96)00048-2. Cited in page 68.

P. Arena, L. Fortuna, G. Muscato, and M. Xibilia. Quaternion algebra. In *Neural Networks in Multidimensional Domains*, volume 234 of *Lecture Notes in Control and Information Sciences*, pages 43–47. Springer London, 1998. Cited in page 35.

L. C. Barros, D. Sanchez, R. Lobo, and E. Esmi. Free mechanical vibrations models via p-fuzzy systems. In *2019 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)*. Atlantis Press, 2019. Cited in page 23.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1023/A:1018054314350. Cited 3 times in page 83, 84, and 89.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. doi: 10.1023/A:1010933404324. Cited 3 times in page 84, 85, and 89.

R. B. Brown. On generalized Cayley-Dickson algebras. *Pacific Journal of Mathematics*, 20(3):415–422, 1967. Cited in page 55.

J. Bruck. On the convergence properties of the hopfield model. *Proceedings of the IEEE*, 78(10):1579–1585, Oct 1990. ISSN 0018-9219. doi: 10.1109/5.58341. Cited in page 25.

S. Buchholz and G. Sommer. Hyperbolic multilayer perceptron. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 129–133, 2000. Cited in page 65.

T. Bülow. Hypercomplex Spectral Signal Representations for Image Processing and Analysis. Technical Report NR 9903, Kiel University, Aug. 1999. Available at: http://www.informatik.uni-kiel.de/en/department-of-computer-science/technical-reports. Cited 3 times in page 35, 51, and 68.

M. Burda. paircompviz: An R Package for Visualization of Multiple Pairwise Comparison Test Results, 2013. Cited in page 90.

F. Z. Castro and M. E. Valle. Continuous-Valued Quaternionic Hopfield Neural Network for Image Retrieval: A Color Space Study. In *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 186–191, Oct 2017a. doi: 10.1109/BRACIS.2017.52. Cited 2 times in page 51 and 68.

F. Z. Castro and M. E. Valle. Continuous-Valued Octonionic Hopfield Neural Network. In *Proceedings Series of the Brazilian Society of Computational and Applied Mathematics. Sociedade Brasileira de Matemática Aplicada e Computacional.*, volume 6, São José dos Campos – Brazil, September 2017b. doi: 10.5540/03.2018.006.01.0344. Cited 2 times in page 25 and 26.

F. Z. Castro and M. E. Valle. Some Remarks on the Stability of Discrete-Time Complex-Valued Multistate Hopfield Neural Networks. In *Proceedings Series of the Brazilian Society of Computational and Applied Mathematics. Sociedade Brasileira de Matemática Aplicada e Computacional.*, pages 010328–1–010328–7, Campinas – Brazil, September 2018. Cited in page 29.

B. Chen, X. Qi, X. Sun, and Y.-Q. Shi. Quaternion pseudo-Zernike moments combining both of RGB information and depth information for color image splicing detection. *Journal of Visual Communication and Image Representation*, 49:283–290, 2017. ISSN 1047-3203. doi: 10.1016/j.jvcir.2017.08.011. Cited in page 68.

T. Chiueh and R. Goodman. Recurrent Correlation Associative Memories. *IEEE Trans. on Neural Networks*, 2:275–284, Feb. 1991. Cited 9 times in page 19, 26, 29, 32, 38, 59, 62, 63, and 97.

T. Chiueh and R. Goodman. *Recurrent Correlation Associative Memories and their VLSI Implementation*, chapter 16, pages 276–287. Oxford University Press, Oxford, U.K., 1993. Cited 4 times in page 19, 20, 25, and 97.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. ISSN 1573-0565. doi: 10.1023/A:1022627411411. URL `https://doi.org/10.1023/A:1022627411411`. Cited in page 33.

N. Cristianini, J. Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000. Cited in page 33.

F. Z. de Castro and M. E. Valle. A broad class of discrete-time hypercomplex-valued hopfield neural networks. *Neural Networks*, 122:54 – 67, 2020. doi: https://doi.org/10.1016/j.neunet.2019.09.040. Cited 11 times in page 20, 29, 55, 56, 57, 58, 63, 67, 68, 72, and 97.

R. Delangue, F. Sommen, and V. Souček. *Clifford algebra and spinor-valued functions : a function theory for the Dirac operator*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1992. ISBN 079230229X. Cited 2 times in page 55 and 57.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. Cited in page 90.

T. A. Ell and S. J. Sangwine. Quaternion involutions and anti-involutions. *Computers and Mathematics with Applications*, 53(1):137–143, 2007. ISSN 0898-1221. doi: 10.1016/j.camwa.2006.10.029. Cited 3 times in page 55, 56, and 57.

A. Ferreira and M. Figueiredo. Boosting Algorithms: A Review of Methods, Theory, and Applications. In C. Zhang and Y. Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pages 35–85. Springer, 2012. doi: 10.1007/978-1-4419-9326-7_2. Cited in page 85.

J. Gan. Discrete Hopfield neural network approach for crane safety evaluation. In *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, pages 40–43, May 2017. doi: 10.1109/ICMSC.2017.7959439. Cited in page 19.

C. García and J. A. Moreno. The Hopfield Associative Memory Network: Improving Performance with the Kernel “Trick”. In *Lecture Notes in Artificial Inteligence - Proceedings of IBERAMIA 2004*, volume 3315 of *Advances in Artificial Intelligence – IBERAMIA 2004*, pages 871–880. Springer-Verlag, 2004a. Cited 2 times in page 33 and 97.

C. García and J. A. Moreno. The Kernel Hopfield Memory Network. In P. M. A. Sloot, B. Chopard, and A. G. Hoekstra, editors, *Cellular Automata*, pages 755–764, Berlin, Heidelberg, 2004b. Springer Berlin Heidelberg. Cited 2 times in page 33 and 97.

G. Garcia and J. A. Moreno. The Hopfield associative memory network: Improving performance with kernel trick. In H. Springer, Berlin, editor, *Advanced in Artificial Intelligence-IBERAMIA 2004*, volume 3315 of *Proc. 9th IBERAMIA*, pages 871–880, July 2004. Cited in page 19.

A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019. Cited 2 times in page 83 and 85.

E. Goles-Chacc, F. Fogelman-Soulie, and D. Pellegrin. Decreasing Energy Functions as a tool for studying threshold networks. *Discrete Applied Mathematics*, 12:261–277, 1985. Cited in page 25.

G. Golub and C. van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 3th edition, 1996. Cited in page 28.

A. B. Greenblatt and S. S. Agaian. Introducing quaternion multi-valued neural networks with numerical examples. *Information Sciences*, 423:326–342, 2018. ISSN 0020-0255. doi: 10.1016/j.ins.2017.09.057. URL http://www.sciencedirect.com/science/article/pii/S0020025516316383. Cited in page 68.

E. R. Hancock and M. Pelillo. A Bayesian interpretation for the exponential correlation associative memory. *Pattern Recognition Letters*, 19(2):149–159, Feb. 1998. Cited 4 times in page 32, 62, 63, and 97.

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990. Cited in page 84.

M. H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA, 1995. Cited in page 18.

M. H. Hassoun and P. B. Watta. The hamming associative memory and its relation to the exponential capacity dam. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 583–587 vol.1, June 1996. doi: 10.1109/ICNN.1996.548960. Cited 3 times in page 62, 63, and 97.

M. H. Hassoun and P. B. Watta. Associative Memory Networks. In E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*, pages C1.3:1–C1.3:14. Oxford University Press, 1997. Cited 3 times in page 19, 27, and 98.

M. H. Hassoun and A. M. Youssef. A New Recording Algorithm for Hopfield Model Associative Memories. In *Neural Network Models for Optical Computing*, volume 882 of *Proceedings of SPIE*, pages 62–70, 1988. Cited in page 30.

S. Haykin. *Neural Networks and Learning Machines*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition edition, 2009. Cited 2 times in page 18 and 27.

D. Hebb. *The Organization of Behavior*. John Wiley & Sons, New York, 1949. Cited in page 27.

A. Hirose. *Complex-Valued Neural Networks*. Studies in Computational Intelligence. Springer, Heidelberg, Germany, 2nd edition edition, 2012. Cited 3 times in page 20, 28, and 97.

M. W. Hirsh. Dynamical Systems. In P. Smolensky, M. Mozer, and D. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum Associates, Publishers, Mahwah, NJ., 1996. Cited in page 19.

E. Hitzer, T. Nitta, and Y. Kuroe. Applications of clifford's geometric algebra. *Advances in Applied Clifford Algebras*, 23(2):377–404, Jun 2013. doi: 10.1007/s00006-013-0378-4. Cited in page 55.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998. Cited 3 times in page 83, 85, and 89.

J. Hopfield and D. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985. Cited in page 19.

J. Hopfield and D. Tank. Computing with neural circuits: A model. *Proceedings of the National Academy of Sciences*, 233:625–633, August 1986. Cited in page 19.

J. J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, Apr. 1982. Cited 4 times in page 18, 19, 26, and 27.

J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81:3088–3092, May 1984. Cited in page 19.

T. Isokawa, H. Nishimura, N. Kamiura, and N. Matsui. Dynamics of Discrete-Time Quaternionic Hopfield Neural Networks. In J. M. Sá, L. A. Alexandre, W. Duch, and D. Mandic, editors, *Artificial Neural Networks – ICANN 2007*, volume 4668 of *Lecture Notes in Computer Science*, pages 848–857. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-74690-4_86. Cited in page 68.

T. Isokawa, H. Hishimura, A. Saitoh, N. Kamiura, and N. Matsui. On the Scheme of Multistate Quaternionic Hopfiels Neural Network. In *Proceedings of Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on advanced Intelligent Systems (SCIS and ISIS 2008)*, pages 809–813, Nagoya, Japan, Sept. 2008a. Cited in page 68.

T. Isokawa, H. Hishimura, N. Kamiura, and N. Matsui. Associative Memory in Quaternionic Hopfield Neural Network. *International Journal of Neural Systems*, 18(02):135–145, 2008b. doi: 10.1142/S0129065708001440. Cited 2 times in page 57 and 68.

T. Isokawa, H. Nishimura, and N. Matsui. Commutative quaternion and multistate Hopfield neural networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2010. doi: 10.1109/IJCNN.2010.5596736. Cited in page 57.

T. Isokawa, H. Nishimura, and N. Matsui. On the fundamental properties of fully quaternionic hopfield network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–4, Brisbane, Australia, June 2012. doi: 10.1109/IJCNN.2012.6252536. Cited in page 25.

T. Isokawa, H. Nishimura, and N. Matsui. Quaternionic Neural Networks for Associative Memories. In A. Hirose, editor, *Complex-Valued Neural Networks*, pages 103–131. Wiley-IEEE Press, 2013. doi: 10.1002/9781118590072.ch5. Cited 3 times in page 20, 37, and 68.

T. Isokawa, H. Yamamoto, H. Nishimura, T. Yumoto, N. Kamiura, and N. Matsui. Complex-valued associative memories with projection and iterative learning rules. *Journal of Artificial Intelligence and Soft Computing Research*, 8(3):237 – 249, 2018. Cited in page 29.

S. Jankowski, A. Lozowski, and J. Zurada. Complex-Valued Multi-State Neural Associative Memory. *IEEE Transactions on Neural Networks*, 7:1491–1496, 1996. Cited 6 times in page 20, 29, 57, 63, 65, and 68.

I. Kanter and H. Sompolinsky. Associative Recall of Memory without Errors. *Physical Review*, 35:380–392, 1987. Cited 2 times in page 19 and 28.

J. Kittler and F. Roli. *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21-23, 2000 Proceedings*. Springer, 2003. Cited in page 84.

M. Kobayashi. Hyperbolic Hopfield Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2), Feb 2013. ISSN 2162-237X. doi: 10.1109/TNNLS.2012.2230450. Cited 2 times in page 57 and 65.

M. Kobayashi. Rotational invariance of quaternionic hopfield neural networks. *IEEJ Transactions on Electrical and Electronic Engineering*, 11(4):516–520, 2016a. ISSN 1931-4981. doi: 10.1002/tee.22269. Cited 3 times in page 36, 38, and 53.

M. Kobayashi. Hyperbolic Hopfield neural networks with four-state neurons. *IEEJ Transactions on Electrical and Electronic Engineering*, 12(3):428–433, 2016b. doi: 10.1002/tee.22394. Cited 2 times in page 65 and 69.

M. Kobayashi. Global hyperbolic Hopfield neural networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E99.A(12):2511–2516, 2016c. doi: 10.1587/transfun.E99.A.2511. Cited in page 65.

M. Kobayashi. Quaternionic Hopfield neural networks with twin-multistate activation function. *Neurocomputing*, 267:304–310, 2017a. doi: 10.1016/j.neucom.2017.06.013. Cited in page 69.

M. Kobayashi. Fixed points of split quaternionic hopfield neural networks. *Signal Processing*, 136:38–42, 2017b. doi: 10.1016/j.sigpro.2016.11.020. Cited in page 69.

M. Kobayashi. Symmetric complex-valued Hopfield neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 28(4):1011–1015, April 2017c. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2518672. Cited in page 29.

M. Kobayashi. Fast recall for complex-valued Hopfield neural networks with projection rules. In *Computational Intelligence and Neuroscience*, volume 2017, pages 1–6, 05 2017d. Cited in page 29.

M. Kobayashi. Twin-multistate commutative quaternion Hopfield neural networks. *Neurocomputing*, 320:150 – 156, 2018a. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.09.023. Cited in page 57.

M. Kobayashi. Dual-numbered Hopfield neural networks. *IEEJ Transactions on Electrical and Electronic Engineering*, 13(2):280–284, 2018b. doi: 10.1002/tee.22524. Cited in page 67.

M. Kobayashi. Hyperbolic Hopfield neural networks with directional multistate activation function. *Neurocomputing*, 275:2217 – 2226, 2018c. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.10.053. Cited in page 65.

M. Kobayashi. Noise robust projection rule for hyperbolic hopfield neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–5, 2019. doi: 10.1109/TNNLS.2019.2899914. Cited in page 65.

M. Kobayashi. Storage capacity of hyperbolic hopfield neural networks. *Neurocomputing*, 369:185 – 190, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.08.064. Cited in page 65.

B. Kosko. Bidirectional Associative Memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:49–60, 1988. Cited 2 times in page 25 and 27.

B. Kosko. Fuzzy associative memory systems. *Fuzzy expert systems*, pages 135–162, 1992. Cited in page 19.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`. Cited in page 51.

D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1180–1188, Red Hook, NY, USA, 2016. Curran Associates Inc. Cited in page 97.

J. Kuiper. *Quaternions and Rotation Sequences: A primer with applications to robotics, aerospace and virtual reality*. Princeton University Press, 1999. Cited in page 35.

Y. Kultur, B. Turhan, and A. Bener. Ensemble of neural networks with associative memory (enna) for estimating software development costs. *Knowledge-Based Systems*, 22(6):395–402, 2009. Cited in page 84.

L. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, 2 edition, 2014. Cited 5 times in page 83, 84, 85, 89, and 98.

Y. Kuroe. *Models of Recurrent Clifford Neural Networks and Their Dynamics*, chapter 6, pages 133–151. Wiley-Blackwell, 2013. doi: 10.1002/9781118590072.ch6. Cited 2 times in page 58 and 67.

Y. Kuroe and H. Iima. A model of Hopfield-type octonion neural networks and existing conditions of energy functions. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4426–4430, July 2016. doi: 10.1109/IJCNN.2016.7727778. Cited in page 57.

Y. Kuroe, S. Tanigawa, and H. Iima. Models of Hopfield-type Clifford neural networks and their energy functions - hyperbolic and dual valued networks. In *Proceedings of the 18th International Conference on Neural Information Processing - Volume Part I*, ICONIP'11, pages 560–569, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24954-9. doi: 10.1007/978-3-642-24955-6_67. Cited 3 times in page 57, 65, and 67.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. Cited in page 18.

D.-L. Lee. Improvements of complex-valued Hopfield associative memory by using generalized projection rules. *IEEE Transactions on Neural Networks*, 17(5):1341–1347, September 2006. Cited 3 times in page 20, 29, and 57.

C. Li, X. Yu, T. Huang, G. Chen, and X. He. A generalized Hopfield network for nonsmooth constrained convex optimization: Lie derivative approach. *IEEE Transactions on Neural Networks and Learning Systems*, 27:1–14, 11 2015. doi: 10.1109/TNNLS.2015.2496658. Cited in page 19.

J. Li, X. Li, B. Huang, and L. Zhao. Hopfield neural network approach for supervised nonlinear spectral unmixing. *IEEE Geoscience and Remote Sensing Letters*, 13(7):1002–1006, July 2016. ISSN 1545-598X. doi: 10.1109/LGRS.2016.2560222. Cited in page 19.

R. A. Lobo and M. E. Valle. Ensemble of binary classifiers combined using recurrent correlation associative memories. In *Brazilian Conference on Intelligent Systems*, pages 442–455. Springer, 2020. Cited in page 23.

R. A. Lobo, J. M. Palma, C. F. Morais, L. d. P. Carvalho, M. E. Valle, and C. L. F. Ricardo Oliveira. A brief tutorial on quadratic stability of linear parameter-varying model for biomathematical systems. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6, Nov 2019. doi: 10.1109/CHILECON47746.2019.8988071. Cited in page 23.

W. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. Cited in page 18.

R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. Venkatesh. The capacity of the Hopfield associative memory. *IEEE Transactions on Information Theory*, 1:33–45, 1987. Cited 2 times in page 19 and 25.

T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui. Quaternionic multistate Hopfield neural network with extended projection rule. *Artificial Life and Robotics*, 21(1):106–111, Mar 2016. doi: 10.1007/s10015-015-0247-4. Cited in page 68.

T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui. Feed forward neural network with random quaternionic neurons. *Signal Processing*, 136:59–68, 2017. doi: 10.1016/j.sigpro.2016.11.008. Cited in page 68.

C. C. E. Morillo, R. A. L. Carrasco, and J. F. da Costa Meyer. Dinamica de hiv e posterior aids. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 6(2), 2018. Cited in page 23.

M. Müezzinoğlu, C. Güzeliş, and J. Zurada. A New Design Method for the Complex-Valued Multistate Hopfield Associative Memory. *IEEE Transactions on Neural Networks*, 14(4): 891–899, July 2003. Cited 2 times in page 29 and 65.

T. Nitta and Y. Kuroe. Hyperbolic gradient operator and hyperbolic back-propagation learning algorithms. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1689–1702, 2018. doi: 10.1109/TNNLS.2017.2677446. Cited 2 times in page 65 and 97.

A. J. Noest. Associative memory in sparse phasor neural networks. *EPL (Europhysics Letters)*, 6(5):469, 1988a. Cited 3 times in page 20, 28, and 63.

A. J. Noest. Discrete-state phasor neural networks. *Physical Review A*, 38:2196–2199, Aug 1988b. doi: 10.1103/PhysRevA.38.2196. Cited 3 times in page 20, 28, and 63.

J. Ontrup and H. Ritter. Text categorization and semantic browsing with self-organizing maps on non-euclidean spaces. In L. De Raedt and A. Siebes, editors, *Principles of Data Mining and*

Knowledge Discovery, pages 338–349, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. Cited in page 65.

G. Pajares, M. Guijarro, and A. Ribeiro. A Hopfield neural network for combining classifiers applied to textured images. *Neural Networks*, 23(1):144–153, Jan. 2010. ISSN 0893-6080. doi: 10.1016/j.neunet.2009.07.019. Cited in page 19.

J. P. Papa, G. H. Rosa, D. R. Pereira, and X.-S. Yang. Quaternion-based Deep Belief Networks fine-tuning. *Applied Soft Computing*, 60:328–335, 2017. ISSN 1568-4946. doi: 10.1016/j.asoc.2017.06.046. Cited in page 68.

T. Parcollet, M. Morchid, and G. Linarès. A survey of quaternion neural networks. *Artificial Intelligence Review*, 2019. doi: 10.1007/s10462-019-09752-1. Cited in page 97.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Cited 2 times in page 89 and 94.

R. Perfetti and E. Ricci. Recurrent Correlation Associative Memories: A Feature Space Perspective. *IEEE Transactions on Neural Networks*, 19(2):333–345, Feb 2008. doi: 10.1109/TNN.2007.909528. Cited 9 times in page 19, 32, 33, 43, 44, 46, 62, 63, and 97.

L. Personnaz, I. Guyon, and G. Dreyfus. Information storage and retrieval in spin glass like neural networks. *Journal of Physics Letter*, 46:L359–L365, 1985. Cited 3 times in page 19, 25, and 28.

R. Polikar. Ensemble Learning. In C. Zhang and Y. Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pages 1–34. Springer, 2012. doi: 10.1007/978-1-4419-9326-7_1. Cited in page 83.

M. P. Ponti Jr. Combining classifiers: from the creation of ensembles to the decision fusion. In *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, pages 1–10. IEEE, 2011. Cited 2 times in page 83 and 84.

C.-A. Popa. Matrix-Valued Hopfield Neural Networks. In L. Cheng, Q. Liu, and A. Ronzhin, editors, *Advances in Neural Networks – ISNN 2016: 13th International Symposium on Neural Networks, ISNN 2016, St. Petersburg, Russia, July 6-8, 2016, Proceedings*, pages 127–134. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-40663-3_15. Cited in page 58.

M. Ravanelli, T. Parcollet, A. Rouhe, P. Plantinga, E. Rastorgueva, L. Lugosch, N. Dawalatabad, C. Ju-Chieh, A. Heba, F. Grondin, W. Aris, C.-F. Liao, S. Cornell, S.-L. Yeh, H. Na, Y. Gao, S.-W. Fu, C. Subakan, R. De Mori, and Y. Bengio. Speechbrain. https://github.com/speechbrain/speechbrain, 2021. Cited in page 20.

R. Rojas. Neural networks: a systematic introduction. *Springer, New York Edwards RE, New J, Parker LE (2012) Predicting future hourly residential electrical consumption: a machine learning case study. Energy Build*, 49:591–603, 1996. Cited in page 27.

F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65:386–408, 1958. Cited in page 18.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. Cited in page 18.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 2002. URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8684. Cited in page 33.

G. Serpen. Hopfield Network as Static Optimizer: Learning the Weights and Eliminating the Guesswork. *Neural Processing Letters*, 27(1):1–15, 2008. doi: 10.1007/s11063-007-9055-8. Cited in page 19.

F. Shang and A. Hirose. Quaternion Neural-Network-Based PolSAR Land Classification in Poincare-Sphere-Parameter Space. *IEEE Transactions on Geoscience and Remote Sensing*, 52:5693–5703, 2014. Cited in page 68.

A. Shenitzer, I. Kantor, and A. Solodovnikov. *Hypercomplex Numbers: An Elementary Introduction to Algebras*. Springer New York, 1989. Cited in page 55.

Y. Song, B. Xing, L. Guo, and X. Xu. System parameter identification experiment based on Hopfield neural network for self balancing vehicle. In *2017 36th Chinese Control Conference (CCC)*, pages 6887–6890, July 2017. doi: 10.23919/ChiCC.2017.8028442. Cited in page 19.

P. Sussner and M. E. Valle. Grayscale Morphological Associative Memories. *IEEE Transactions on Neural Networks*, 17(3):559–570, May 2006. Cited in page 19.

D. E. Sánchez, J. M. Palma, R. A. Lobo, J. F. C. A. Meyer, C. F. Morais, A. Rojas-Palma, and R. C. L. F. Oliveira. Modeling and stability analysis of salmon mortality due to microalgae bloom using linear parameter-varying structure. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6, Nov 2019. doi: 10.1109/CHILECON47746.2019.8987679. Cited in page 23.

G. Tanaka and K. Aihara. Complex-Valued Multistate Associative Memory With Nonlinear Multilevel Functions for Gray-Level Image Reconstruction. *IEEE Transactions on Neural Networks*, 20(9):1463–1473, Sept. 2009. doi: 10.1109/TNN.2009.2025500. Cited in page 29.

M. Valle. Complex-Valued Recurrent Correlation Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(9):1600–1612, September 2014a. doi: 10.1109/TNNLS.2014.2341013. Cited 9 times in page 20, 25, 26, 32, 38, 58, 59, 63, and 71.

M. E. Valle. A Novel Continuous-Valued Quaternionic Hopfield Neural Network. In *Proceedings of the Brazilian Conference on Intelligent Systems 2014 (BRACIS 2014)*, São Carlos, Brazil, October 2014b. Cited 3 times in page 36, 37, and 69.

M. E. Valle. Quaternionic Recurrent Correlation Neural Networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2018. doi: 10.1109/IJCNN.2018.8489714. Cited 9 times in page 20, 32, 34, 38, 58, 59, 62, 69, and 71.

M. E. Valle and F. Z. Castro. On the Dynamics of Hopfield Neural Networks on Unit Quaternions. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2464–2471, June 2018. doi: 10.1109/TNNLS.2017.2691462. Cited 3 times in page 37, 57, and 69.

M. E. Valle and R. A. Lobo. An introduction to quaternion-valued recurrent projection neural networks. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 848–853. IEEE, 2019. Cited in page 23.

M. E. Valle and R. A. Lobo. Quaternion-valued recurrent projection neural networks on unit quaternions. *arXiv preprint arXiv:2001.11846*, 2020. Cited in page 23.

M. E. Valle and R. A. Lobo. Hypercomplex-valued recurrent correlation neural networks. *Neurocomputing*, 432:111 – 123, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.12.034. URL http://www.sciencedirect.com/science/article/pii/S0925231220319342. Cited in page 23.

J. R. Vallejo and E. Bayro-Corrochano. Clifford Hopfield Neural Networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3609–3612, June 2008. doi: 10.1109/IJCNN.2008.4634314. Cited 2 times in page 57 and 67.

M. Van Erp, L. Vuurpijl, and L. Schomaker. An overview and comparison of voting methods for pattern recognition. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 195–200. IEEE, 2002. Cited in page 85.

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. Cited 2 times in page 89 and 94.

J. Vaz and R. da Rocha. *An Introduction to Clifford Algebras and Spinors*. Oxford University Press, 2016. Cited 2 times in page 55 and 57.

Q. Wang, W. Shi, P. M. Atkinson, and Z. Li. Land cover change detection at subpixel resolution with a Hopfield neural network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(3):1339–1352, March 2015. ISSN 1939-1404. doi: 10.1109/JSTARS.2014.2355832. Cited in page 19.

T. Weise and R. Chiong. An alternative way of presenting statistical test results when evaluating the performance of stochastic approaches. *Neurocomputing*, 147:235–238, 2015. doi: 10.1016/j.neucom.2014.06.071. Cited in page 90.

Y. Xia, C. Jahanchahi, and D. Mandic. Quaternion-Valued Echo State Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 26:663–673, 2015. Cited in page 68.

L. Xiaodong, L. Aijun, Y. Changjun, and S. Fulin. Widely Linear Quaternion Unscented Kalman Filter for Quaternion-Valued Feedforward Neural Network. *IEEE Signal Processing Letters*, 24(9):1418–1422, Sept 2017. doi: 10.1109/LSP.2017.2734652. Cited in page 68.

D. Xu, Y. Xia, and D. P. Mandic. Optimization in Quaternion Dynamic Systems: Gradient, Hessian, and Learning Algorithms. *IEEE Transactions on Neural Networks and Learning Systems*, 27(2):249–261, Feb 2016. doi: 10.1109/TNNLS.2015.2440473. Cited in page 68.

C. Zhang and Y. Ma, editors. *Ensemble Machine Learning: Methods and Applications*. Springer, 2012. doi: 10.1007/978-1-4419-9326-7. Cited 3 times in page 83, 84, and 98.

H. Zhang, Y. Hou, J. Zhao, L. Wang, T. Xi, and Y. Li. Automatic welding quality classification for the spot welding based on the Hopfield associative memory neural network and chernoff face description of the electrode displacement signal features. *Mechanical Systems and Signal Processing*, 85:1035 – 1043, 2017. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2016.06.036. Cited in page 19.

——————————————————————-

APPENDIX A- Remarks on the Computational Implementation of HRCNNs

In our codes, a hypercomplex-valued vector $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]$, where $x_i(t) = x_{i0} + x_{i1}\mathbf{i}_1 + \ldots + x_{in}\mathbf{i}_n$ for all $i = 1, \ldots, N$, is represented by a $(N, n+1)$-array of real numbers:

$$\mathtt{x} = \begin{bmatrix} x_{10} & x_{11} & \ldots & x_{1n} \\ x_{20} & x_{21} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N0} & x_{N1} & \ldots & x_{Nn} \end{bmatrix}.$$

Similarly, the fundamental memory set $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^P\}$ is represented by a $(N, n+1, P)$-array `U` such that `U`$(:,:,\xi)$ corresponds to $\mathbf{u}^\xi$, for $\xi = 1, \ldots, P$. An arbitrary hypercomplex-valued ECNN is implemented as a function which receives the following inputs and outputs the final vector state as well as an array with the evolution of the function E given by (3.17):

1. The symmetric bilinear form $\mathcal{B}$ and its parameters.
2. The activation function ϕ and its parameters.
3. The array `U` corresponding to the fundamental memory set.
4. The array `x` corresponding to the input vector $\mathbf{x}(0)$.
5. The optional parameters α and β as well as the maximum number of iterations.

Note that both hypercomplex-valued product, as well as the reverse-involution, are implicitly defined in the symmetric bilinear form $\mathcal{B}$ which is passed as an argument to the hypercomplex-valued ECNN model. Finally, we would like to point out that there is one code for the synchronous hypercomplex-valued ECNN model and another code for the asynchronous version. To fasten the implementation of the asynchronous hypercomplex-valued ECNN, instead of using (3.16), we updated the weights $w_\xi(t+1)$ as follows for $\xi = 1, \ldots, P$: Suppose only the ith neuron have been updated at time t, that is, $x_j(t+1) = x_j(t)$ for all $j \neq i$. Then, the weight $w_\xi(t+1)$ satisfies

$$w_\xi(t+1) = w_\xi(t) \exp\Big\{\alpha\mathcal{B}\big(u_i^\xi, x_i(t+1) - x_i(t)\big)\Big\}. \tag{1}$$

Note that, in contrast to (3.16) which evaluates N-times the symmetric bilinear form $\mathcal{B}$, (1) requires a single evaluation of $\mathcal{B}$. Recall that the jupyter notebook of this experiment, implemented in `Julia language`, can be found in `https://github.com/mevalle/Hypercomplex-Valued-Recurrent-Correlation-Neural-Networks`.