Noise Suppression By Artificial Neural Networks

Thesis submitted in partial
fulfilment of the
requirements for the degree
of

Master of
Science in
Electronics and Communication Engineering by Research

by

Aman Singh
201331185
aman.singh@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA

JUNE 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled NOISE SUPPRESSION BY ARTIFICAL NEURAL NETWORK by AMAN SINGH, has been carried out under my supervision and is not submitted elsewhere for a degree.

29- June-2023

Date

Adviser: Rama Murthy Garimella

To My Mother and Sister

Acknowledgments

My journey at IIIT-Hyderabad has been a truly remarkable experience, filled with both highs and lows. Throughout this journey, the successful outcome of my thesis has been made possible thanks to the unwavering support and encouragement of numerous individuals. As I prepare to submit my MS thesis, I would like to express my heartfelt gratitude to all those who have played a significant role in helping me reach this milestone.

First and foremost, I extend my deepest appreciation to my thesis advisor, Dr. Rama Murthy Garimella, for accepting me as their student and for providing invaluable guidance, knowledge, and support throughout my MS journey. Their unwavering assistance during challenging times has been instrumental in strengthening my resolve and keeping me motivated to push further.

I would also like to express my gratitude to my lab mates in SPCRC and my friends at IIIT for creating a positive work environment that fostered collaboration and growth. I am especially grateful to individuals like Moneish Kumar, Sikander Sharda, Aaditya M Nair, Shubham Agarwal, and Saksham Agrawal, who have not only been sources of support but have also contributed to the fun-filled moments and cherished memories at IIIT.

Moreover, I would like to acknowledge that the constant support and understanding of my parents have been crucial in my academic accomplishments. Their unwavering belief in me has been a constant source of strength and motivation.

Lastly, I am grateful to the entire IIIT community for providing an inspiring and nurturing environment that has allowed me to thrive and grow. The abundance of opportunities I have been given has contributed significantly to my personal and academic development.

With deep gratitude and a sense of accomplishment, I look forward to the next chapter of my journey, knowing that the experiences and relationships cultivated at IIIT will continue to shape my future.

Abstract

In the research field of Artificial Neural Networks, there are ongoing efforts to address the issue of noise corruption in patterns. Various approaches and techniques have been explored to suppress noise and improve the accuracy of pattern classification.

One of the research themes in this direction is the theory of Support Vector Machines [52] (SVM). SVMs aim to find an optimal hyperplane that maximizes the margin between classes, effectively reducing the impact of noise on classification. This approach formulates the problem of noise suppression as a quadratic programming problem.

Hopfield Associative Memory [5] is another research area that has been investigated for noise suppression. The concept of null vectors of perceptron’s, which refers to vectors in the null space of the perceptron's transformation matrix, is introduced and utilized to suppress noise. By studying the null vectors of Extreme Machine Learning [49], researchers have explored their potential for noise suppression in pattern classification tasks.

In addition to these approaches, the idea of stacking associative memories has been utilized to develop Hopfield neural networks [5] capable of suppressing noise in different dimensions. This includes 1-D, 2-D, and 3-D Hopfield neural networks that are designed to effectively handle noise corruption in patterns of various dimensions.

Furthermore, to further enhance classification accuracy by suppressing noise, a real-valued Hopfield neural network based on a novel model of artificial neuron called the ceiling neuron has been proposed and studied. This innovation aims to improve noise suppression capabilities and overall performance in pattern classification tasks.

Overall, this thesis explores and innovates different approaches to suppress noise in pattern classification using various neural network models, such as Support Vector Machines, Hopfield Associative Memory, and real-valued Hopfield neural networks based on ceiling neurons. These efforts contribute to the ongoing research in noise suppression techniques and aim to improve the accuracy and reliability of pattern classification systems.

Contents

List of Figures

Figure Page

Chapter-1
Introduction

Since the dawn of civilization, Homo sapiens have endeavored to understand nature and have devised various theoretical and experimental tools to discover its hidden laws. Mathematical tools have provided the basis for modeling both natural and artificial phenomena. In the field of Signal Processing, specifically, analytical tools known as filters have been developed to extract one-, two-, or three-dimensional signals corrupted by noise.

Frequency-selective filters, such as lowpass, high-pass, and bandpass filters, have been highly successful in removing noise signals that exhibit frequency selectivity. Moreover, in cases where the noise is a non-deterministic signal (modeled as a wide-sense stationary random process), techniques like the Wiener filter and Kalman filter have proven to be extremely effective in extracting the desired signal from the noise, especially in the presence of white noise. Therefore, the problem of noise removal plays a crucial role in signal processing.

In the field of Artificial Neural Networks (ANNs), researchers have made significant advancements in the development of Associative memories, which effectively remove noise from input pattern vectors. Additionally, in the domain of error correcting code design, several codes have been proposed by researchers that are capable of detecting and correcting errors in input information vectors corrupted by noise. Interestingly, it has been recognized that error correcting codes and associative memories share a natural relationship.

Effectively, the existing models of Artificial Neural Networks (ANNs) do not consider the presence of noise in pattern vectors for classification purposes. To the best of our knowledge, one of the earliest attempts to address this issue was proposed in the paper titled "HYBRID NEURAL NETWORKS." In this work, a Hybrid Neural Network architecture was introduced, combining a Hopfield Associative Memory (HAM) with a Multilayer Perceptron [51] (MLP) for classification tasks. By incorporating the Hopfield Neural Network (HNN) as either an input or output component of the MLP, the system effectively removes noise from the input and/or output. The findings of this research paper served as a significant motivation for the current thesis.

Let's summarize the evolution of noise suppression approaches from their historical origins. The quest to determine the orbit of Sirius through astronomical measurements led Gauss to propose the method of least mean square estimation. This method eventually developed into what is known as the Linear Regression approach. Huber introduced a function to mitigate the impact of outliers in data. Engineers were particularly interested in removing unwanted noise that corrupts signals, which gave rise to the fields of analog and digital signal processing.

Both analog and digital filters were designed to tackle frequency-selective noise, employing techniques such as low-pass, high-pass, band-pass, and notch filters. By considering noise as a random or non-deterministic signal, Wiener

proposed the Minimum Mean Square Error (MMSE) filter, which found numerous applications. Kalman utilized the state space representation of linear systems and developed the Kalman filter, which has been widely utilized in various applications.

Nonlinear filters, including the median filter, were introduced to counter the effects of noise in one-dimensional, two-dimensional, and three-dimensional signals. In communication applications, error-correcting codes were successfully designed by introducing redundancy, mapping an information vector into a codeword. These codes proved effective in correcting errors in signals received by the receiver.

Hopfield proposed a model of Artificial neural network that can emulate the associative memory routinely utilized in biological memories. Hopfield neural network (HNN) maps a noise corrupted input vector into a stable state lying on the symmetric unit hypercube.

Chapter-2

Noise Suppression in Artificial Neural Network

1. Introduction

Models of Artificial Neural Networks (ANNs) have been developed to emulate the capabilities of the biological brain. Single Layer Perceptron [51] (SLP) and Multi-Layer Perceptron [51] (MLP) have proven to be highly successful ANNs for classification tasks using supervised learning. In MLP, the number of hidden layers is typically kept small, as it serves the purpose of classification in many applications.

In the pursuit of training ANNs to recognize handwritten characters, Yann LeCun introduced the fundamental concepts of Convolutional Neural Networks [50] (CNNs). CNNs typically have a larger number of layers of neurons compared to earlier MLP-based models. This learning paradigm was coined as "Deep Learning" to distinguish it from previous approaches. Yann LeCun successfully derived the back-propagation algorithm for CNNs.

One of the authors proposed the concept of a dynamic synapse modelled as a Finite Impulse Response (FIR) filter [2]. ANNs based on this model were developed by the author and other researchers. It is evident from the theory of linear filtering that such a dynamic synapse has the capability to suppress frequency-selective noise, such as low-pass filtering. To the authors' knowledge, the noise suppression abilities of CNNs have not been investigated by earlier researchers. This research aims to explore that aspect. Furthermore, the noise suppression capabilities of arbitrary MLPs and Extreme Learning Machines [49] (ELMs) have not been investigated either. Bruck et al. explored the connection between Hopfield Associative Memory [5] (HAM) and generalized neural networks with error-correcting codes, examining the noise suppression abilities of HAM. This work inspired the authors to investigate the noise suppression abilities of ANNs.

This chapter is structured as follows: Section 2 provides a summary of the existing literature on the noise suppression abilities of associative memories, such as the Hopfield neural network. Sections 3, 4, 5, and 6 discuss the noise suppression abilities of CNNs, MLPs, ELMs, and SLPs, respectively, based on the concept of null vectors.

2. Review of Literature

2.1 Associative Memories: *Noise Suppression*

Bruck et al. demonstrated that the Hopfield neural network has a natural association with a graph-theoretic code. This means that the local optimum of the quadratic energy function is linked to the codewords. In a more general sense, the local optimum of a higher degree energy function in a generalized neural network corresponds to the codewords of an error-correcting code. Each codeword of a linear or nonlinear error-correcting code represents a stable state, and each stable state of a generalized neural network represents a codeword [1]. Therefore, when patterns are quantized into a finite number of values (e.g., 0, 1, 2, ..., 'p' where 'p' is a prime number), the error-correcting code associated with a generalized neural network or associative memory performs "clustering" and facilitates noise suppression. In other words, it corrects a certain number of errors based on the minimum distance of the code. As a result, such associative memories enable noise suppression.

Note 1: We expect connection between CNNs and convolutional codes.

2.2 Hybrid ANNs: Noise Suppression

In [3], the first author introduced the concept of hybrid neural networks, which involve cascading an associative memory/encoder (based on an error-correcting code) with an MLP. This configuration allows the hybrid ANNs to perform clustering either before or after classification using the MLP. The paper also discussed the idea of a hybrid neural network, which combines both feedforward and feedback neural networks with an encoder positioned before and/or after the MLP. As mentioned earlier, hybrid neural networks have the ability to perform noise suppression both before and/or after classification by the MLP or CNN.

2.3 Motivation for Noise suppression in MLP

In [3], the first author introduced hybrid neural networks by combining an associative memory/encoder (based on an error-correcting code) with an MLP. This combination can be positioned either before or after the MLP block. These hybrid ANNs have the capability to perform clustering either before or after classification by the MLP. Additionally, the paper also discussed hybrid neural networks that involve both feedforward and feedback neural networks, along with an encoder positioned before and/or after the MLP. As mentioned earlier, hybrid neural networks provide noise suppression both before and/or after classification by the MLP or CNN, as previously discussed.

3. Linear Convolutional Layers: Finite Dimensional Linear Transformations

In this section, we will explore the transformations performed by convolutional layers in a Convolutional Neural Network (CNN). We are motivated by the fact that linear filtering of the input signal, after being trained, corresponds to the convolution of the input with the impulse response of a Linear Time Invariant (LTI) system.

Let's consider a layer with N neurons. The outputs of these neurons are combined into an N-dimensional column vector, denoted as $\bar{U}$ (a vector of size $N \times 1$). These neuron outputs are then convolved or

correlated with a set of mask coefficients. Let's assume the mask has a length of M, denoted as $\bar{H}$ = [H(0), H(1), ..., H(M-1)]. The input vector $\bar{U}$ is convolved or correlated with the mask vector $\bar{H}$. The resulting output vector, denoted as $\bar{Y}$, has a length of M + N - 1.

For the purpose of illustration, let's consider a scenario where N = 6 and M = 3. The convolution or correlation of the input vector $\bar{U}$ with the mask vector $\bar{H}$ effectively corresponds to a finite-dimensional linear transformation. In other words, we can express this transformation as $\bar{Y}$ = $\bar{H}$ $\bar{U}$, where $\bar{H}$ is generally a (M + N - 1) × N matrix.

$\acute{Y}$ = (y(1) y(2) ... y(8))

$\acute{U}$ = (u(1) u(2) ... u(6))

h(0) ≠ 0

$\acute{H}$=¿

In convolution even though $\bar{Y}$ is a vector of length 8, only first 4 components are considered (since last 4 components do not involve correlation with all the mask coefficients i.e., only some mask coefficients are involved). It can be readily noted that $\bar{H}$ is a rectangular Toeplitz matrix.

Note 2: The generalization to arbitrary dimensional input vector and mask vector is straight forward, avoided for brevity.

It should be noted that if pooling/sub-sampling operation is “linear” (for instance “average” computation or down sampling), then the output of pooling vector $\bar{Y}$ corresponds to a finite dimensional linear transformation, $\bar{P}$.

$$\acute{Z} = \acute{P}\acute{Y} = \acute{P}\acute{H}\acute{U} \quad (1)$$

Note 3: Since only first four rows of $\bar{H}$ (8 × 6 matrix $\bar{H}$) are considered to determine the output $\bar{Y}$, it is clear that we have $\hat{Y}$ =$\hat{H}$ $\bar{U}$ (where $\hat{H}$ is 4x6 matrix). Thus, the null space of $\hat{H}$ is of dimensional 2. In other words, non-trivial input $\hat{U}$ exists such that $\hat{H}$ $\hat{U}$ =0.

Hence, we have

$$\hat{Y} = \hat{H}(\bar{U} + \hat{U}) = \hat{H}\,\bar{U} \quad (2)$$

Equivalently vectors in the null space of $\hat{H}$ are the noise vectors which will be suppressed by the trained convolutional layer in a CNN. Thus, we are able to quantify certain noise suppression ability of a CNN. In general, the null space ($\hat{H}$) is of dimension N-M-1. All such noise vectors are suppressed by a trained convolutional layer. $\hat{H}$ is also a Toeplitz matrix.

= [:], where

$$\acute{A} = \begin{bmatrix} h(0) & h(1) & h(2) & 0 \\ 0 & h(0) & h(1) & h(2) \\ 0 & 0 & h(0) & h(1) \\ 0 & 0 & 0 & h(0) \end{bmatrix}$$

i.e. $\acute{A}$, being upper triangular is non-singular if h(0)≠0.

Some properties of $\acute{A}$:

1. Eigenvalues of $\acute{A}$ are all equal to “h(0).”

2. The null space dimension is ‘4′ if h(0) =0 and is ‘0 ′ if h(0)≠0.

The columns of $\acute{B}$ (2 of them) are linearly dependent on the columns of $\acute{A}$. In fact, if

$$\acute{B}=\begin{bmatrix}\acute{b}_1 & : & \acute{b}_2\end{bmatrix} \quad (3)$$

$$\acute{A}\acute{c}_1=\acute{b}_1 \quad (4)$$

$$\acute{A}\acute{c}_2=\acute{b}_2 \quad (5)$$

$$\acute{c}_1=\acute{A}^{-1}\acute{b}_1 \quad (6)$$

$$\acute{c}_2=\acute{A}^{-1}\acute{b}_2 \quad (7)$$

Note 4 Upper triangular Toeplitz matrix can be effectively inverted [4].

Example:

$$\acute{A}=\begin{bmatrix}h(0) & h(1) & h(2) & 0\\ 0 & h(0) & h(1) & h(2)\\ 0 & 0 & h(0) & h(1)\\ 0 & 0 & 0 & h(0)\end{bmatrix}$$

Let’s take h(0) to be 1, h(1) to be 0.5 and h(2) to be 0.3.

Then,

$$\acute{A}=\begin{bmatrix}1 & 0.5 & 0.3 & 0\\ 0 & 1 & 0.5 & 0.3\\ 0 & 0 & 1 & 0.5\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$\acute{A}^{-1}$ can be calculated using technique provided in [4]. $\acute{A}^{-1}$ is also an upper triangular Toeplitz matrix.

$$\acute{A}^{-1}=\begin{bmatrix}1 & -0.5 & -0.05 & 0.175\\ 0 & 1 & -0.5 & -0.05\\ 0 & 0 & 1 & -0.5\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Goal: precise determination of noise suppression ability of a trained convolutional layer i.e. null space of $\widehat{H}$ i.e., $\acute{f}_1$, $\acute{f}_2$ are the basis of null space of $\widehat{H}$, then the set of noise vectors which are suppressed by the convolutional layer can be characterized

$\acute{g} = \alpha\acute{f}_1 + \beta\acute{f}_2$, with α, β ∈ R.
(8)

Note 4: Each convolutional layer receives input from the previous layer. Thus, each layer provides certain amount of noise immunity.

Finite Dimensional Linear Filtering: Each convolutional/pooling layer removes noise in a certain linear space/subspace.

Problem Formulation: To determine the basis of null space of $\widehat{H}$.

$\widehat{H}=[\acute{A}:\acute{b}_1:\acute{b}_2]$ with $\acute{A}$ being upper triangular non-singular Toeplitz matrix. $\acute{A}\acute{c}_1=\acute{b}_1$, and $\acute{A}\acute{c}_2=\acute{b}_2$. Further, we have that

$$\acute{b}_1 = \begin{bmatrix} 0 \\ 0 \\ h(2) \\ h(1) \end{bmatrix} \quad \text{and} \quad \acute{b}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ h(2) \end{bmatrix}$$

Suppose, h(2)≠0, h(1)≠0.

Then $\acute{b}_1$, $\acute{b}_2$ are linearly independent. Thus, the basis of null space of $\overline{H}$ can easily be determined. Hence noise suppression ability of CNN can be determined.

Specifically, we have

$$\begin{bmatrix} c_1 \\ -1 \\ 0 \end{bmatrix} \equiv \bar{0} \qquad \text{and} \qquad \begin{bmatrix} c_2 \\ 0 \\ -1 \end{bmatrix} \equiv \bar{0} \ .$$

4. Null Vectors of Artificial Neural Networks

We have previously examined the vectors that lie in the null space of the finite-dimensional linear transformation associated with a Convolutional Neural Network (CNN). We have observed that such vectors are suppressed by a trained CNN. We now realize that this idea also has implications for a trained multi-layer perceptron.

In a multi-layer perceptron, the input from one layer to the next undergoes a linear transformation. Let's consider a scenario where there are 'M' neurons in one layer and their outputs are fed to 'N' neurons in the next layer. This transformation can be captured by a finite-dimensional linear transformation matrix, denoted as T, which is an N x M matrix. In the context of neuron layers, we can define the concept of null vectors associated with this linear transformation.

Let's consider a finite-dimensional linear transformation matrix, $\overline{\mathrm{T}}$, which is associated with the synaptic weights from one layer to the next.

Definition: The null vectors of a layer in a Multi-Layer Perceptron (MLP) are the vectors that belong to the null space of the linear transformation matrix, $\bar{T}$, associated with that layer.

Then a signum operation is applied to the weighted input at each of the 'N' neurons. As a result, the vectors that lie in the null space of T are suppressed. In other words, let $\bar{U}$ be the output vector of a layer and $\bar{Y}$ be obtained as $\bar{Y} = \bar{T}\bar{U}$. If there exists a non-zero vector $\bar{V}$ (belonging to the null space of T) such that

$$\acute{T}\acute{V}=0, \qquad \text{then}$$

$$\acute{T}(\acute{U}+\acute{V}) = \acute{T}\acute{U}=\acute{Y} \tag{9}$$

(If $\acute{V}$ is a noise vector, it is suppressed from propagating to next layer).

Even in a fully trained MLP, where the transformation matrices have a non-empty null space, effective noise removal occurs. This implies that in the context of supervised learning by MLP, once the training is completed and the synaptic weights are frozen, "noisy" patterns (those vectors that belong to the null spaces of the transformation matrices) are effectively suppressed.

5. Null vectors of Extreme learning machine

It should be noted that Extreme learning machine (ELM) [49] is one type of Multilayer Perceptron. In ELM weights from Input layer to Hidden layer are assigned randomly and Hidden layer to Output layer are computed by a closed form expression. Input vector is $\acute{U}$ and $\overline{Y_1}$, $\overline{Y_2}$ are the outputs of layer one, two respectively.

$$\overline{Y_1}=\overline{T_1}\acute{U} \tag{10}$$

$$\overline{Y_2}=\overline{T_2}\overline{Z_1} \tag{11}$$

$\overline{Y_1}$ is output of the first layer of neurons before activation function is applied and $\overline{Z_1}$ is output of first layer after activation function is applied.

$\overline{T_1}$ and $\overline{T_2}$ are finite dimensional linear transformations at first and second layers respectively. i.e., Null Spaces of $\overline{T_1}$, $\overline{T_2}$ determine the “noise” which is suppressed.

Note 5: There is no training in ELM. Null spaces of $\overline{T_1}$ and $\overline{T_2}$ determine the perturbations (could be noise) to $\acute{U}$, $\overline{Z_1}$ which will be suppressed/zeroed out.

From the training data available, it could be determined if the null vectors are like “NOISE” which should be suppressed or they have desired pattern vector information (which should not be suppressed), this holds true for any ANN.

Note 6: Given “NULL VECTOR” information of an arbitrary ANN, can a choice of initial synaptic weights be intelligently made (w.r.t null spaces of linear transformation) such that training time may be reduced.

Note 7: This idea may be of specific importance in design of ELM. i.e., choice of synaptic weights from input to the hidden layer (single one). From the null space of $\overline{T_1}$, it is clear that certain patterns are suppressed/zeroed out.

Cases of Interest: $\overline{Y_1}=\overline{T_1}\acute{U}$ where $\overline{T_1}$ is an 'MXN' matrix.

1. M > N: Null space of $\overline{T1}$ is of interest for us.
2. M = N: i.e., number of input neurons in layer one is same as number of neurons in next layer.
3. M < N: i.e., number of input neurons is smaller than number of neurons in next layer.

Suppose, in the case of ELM, certain "DESIRED" pattern vectors are getting suppressed because they lie in null space of finite dimensional linear transformation/matrix i.e., $\overline{T_1}$. Then, such vectors will not affect the training process.

Note 8: In the case of ELM, there is no backpropagation for changing the weights with whatever initialization of weights we choose from input neurons to neurons in hidden layer, the other weights are computed in closed form.

Suppose the NULL SPACE of $\overline{T_1}$ is empty. Then, none of the input noise vectors are suppressed. Then the entire input space effects the performance of ELM.

6. Single Layer Perceptron: Null Vectors

Goal: Classification of Linearly Separable Patterns, i.e., Training by perceptron convergence law is associated with finitely many transformation matrices, during convergence of learning law.

Lemma: Linear transformation preserves linear separability.

Proof: Refer [5] ... If the weights from inputs to the neurons in a single layer are updated in successive iterations, the finite dimensional linear transformation/ matrix is changing. i.e., rows of linear transformation/matrix correspond to the hyperplanes separating the classes.

In finitely many steps, the synaptic weight matrices are converging to a matrix which properly classifies the patterns.

Note 9: In case, there are maximum possible number of classes (i.e., number of neurons in the single layer + 1 e.g., M hyperplanes that are parallel corresponding to 'M' classes) the rows of successive synaptic weight matrices (on learning) converge to a rank-one matrix.

Note 10: When the number of classes is maximum and the converged synaptic weight matrix is rank-one matrix (i.e., hyperplanes are parallel), the null space has dimension 'M−1', where 'M' is the number of neurons in SLP. The converged synaptic weight matrix, is a

1. Square matrix if the number of inputs is equal to the number of neurons in SLP.

2. Rectangular matrix if the number of inputs is greater than number of neurons in SLP.

7. Noise Suppression in SVM

Consider two classes of patterns that are linearly separable. Rosenblatt demonstrated that the perceptron learning law converges to a hyperplane that correctly classifies all training patterns. It is important to note that this hyperplane, which achieves proper classification, is not unique.

Vapnik introduced the concept of "margin" and formulated the problem of finding the optimal classifying hyperplane (maximizing the margin) as a quadratic programming problem. In cases where patterns are not linearly separable, a kernel function is used to project the patterns into a higher-dimensional space where they become linearly separable. In this higher-dimensional space, the optimal hyperplane is computed to classify the linearly separable patterns.

However, the current design of Support Vector Machines (SVMs) does not fully consider the classification of patterns that are corrupted by noise. Therefore, there is a need for a robust SVM design that can handle classification in the presence of noise. Such an approach requires the specification of a noise model. We propose an approach for robust SVM design and introduce an interesting noise model.

Let's consider the L^2-norm of a noise vector that corrupts the pattern vectors. We propose the following modification: Instead of considering only one optimal hyperplane, we locate a robust optimal hyperplane that is parallel to the hyperplanes with respect to both classes and lies exactly in the middle of them.

If we consider both the norm and direction of a noise vector, we find that noise patterns corrupting patterns of one class that are perpendicular or orthogonal to the optimal hyperplane and point away from the "other" class can have any L^2-norm and still not affect the classification of patterns.

Similarly, noise patterns corrupting patterns of either class that are parallel to the optimal hyperplane can have any L^2-norm and still not affect the classification of patterns.

Noise that corrupts patterns can occur in any direction in a finite-dimensional vector space. To address this, we select an orthonormal basis for the N-dimensional pattern space, with one basis vector being perpendicular to the optimal hyperplane provided by Vapnik's quadratic optimization algorithm (Gram-Schmidt's orthogonalization procedure can be used to choose such an orthonormal basis).

The goal is to quantify the noise suppression ability of SVMs. We use P to denote an orthonormal matrix, where the rows and columns form an orthonormal basis, and N represents the noise vector that corrupts the patterns.

P = ($P_1 : P_2 : \ldots P_N$)

Each P_i is Column vector for each i.

$$P_i^T P_j = \begin{cases} 0 if\ i = j \\ 1 if\ i \neq j \end{cases}$$

N = P C

Components of C quantify the ability of noise to corrupt patterns. Component w.r.t P_1 corresponds to minimum L^2-norm /length which will garble the patterns.

8. Conclusion

In this chapter, we delve into the noise suppression abilities of trained Convolutional Neural Networks (CNNs). We employ the effective idea to characterize the noise suppression capabilities of Single Layer Perceptron (SLP), Multi-Layer Perceptron (MLP), and Extreme Learning Machine (ELM). Furthermore, we review the noise suppression ability of Hopfield associative memory. Additionally, we propose Hybrid neural networks as a means to suppress noise.

Chapter-3

1-D/2-D/3-D Hopfield Associative Memories

1. Introduction

In an attempt to model biological memory, Hopfield proposed the Discrete Time Hopfield Neural Network (DTHNN), which serves as an Associative Memory (AM). The AM is based on a vector of {+1, -1} as the state of the dynamical system, allowing for the storage of one-dimensional information. However, there is a need to design associative memories capable of storing two-dimensional or three-dimensional information.

In a study [13], the author presented the design of Multi-Dimensional Neural Networks. Researchers explored various approaches to designing associative memories that can handle multi-dimensional information. Some attempts involved using {+1, -1} arrays as the system's state, while others explored associative memories with multi-state neurons that have more than two elements [16], [14]. These efforts have numerous applications, including content-based image retrieval and other related problems.

This chapter focuses on proposing methods for the storage and retrieval of 1-D, 2-D, and 3-D information using the principles of associative memory. The chapter's organization is as follows:

- Section 2 provides a review of relevant research literature.
- Section 3 introduces an architecture based on the Parallel Hopfield Neural Network.
- Section 4 proposes new architectures by stacking parallel Hopfield Associative Memories to convert lower dimensions to higher dimensions. It also discusses the Ceiling Neuron based Associative Memory.
- Section 5 presents a novel deep learning architecture.
- Section 6 explores applications of the proposed architectures.
- Section 7 discusses the results obtained from implementing the proposed architectures on black and white images.
- Finally, Section 8 concludes the chapter.

2. Review of Related Literature

In the research literature on Associative Memories, it has been widely recognized that the Hopfield Associative Memory has certain limitations in terms of its state space. To overcome these limitations, researchers have proposed alternative approaches, such as multi-state neuron-based associative memories. Many of these efforts involve using vector-based states for the neural network or dynamical system.

In [13], the author introduced the design of multidimensional associative memories. Furthermore, the convergence theorem of multi-dimensional Hopfield neural networks was proven [8]. These multi-dimensional associative memories, including 2-D and 3-D versions, have found numerous applications in storing images and video data [7].

In this chapter, our goal is to propose simple models of 2-D or 3-D associative memories, inspired by Hopfield's original work. While acknowledging the

limitations of Hopfield's approach, we aim to explore alternative models that can handle multi-dimensional data and serve as associative memories.

3. Parallel Hopfield Neural Network

In this section, we introduce a variation of the Discrete Time Hopfield Neural Network (DTHNN) architecture. Our proposed architecture is inspired by the concept of the CEILING NEURON, which was originally proposed in [9]. The key idea behind this variation is the introduction of multiple thresholds at each neuron, as opposed to a single threshold.

By incorporating this idea, we formulate a nonlinear dynamical system that serves as a two-dimensional associative memory. This modified architecture allows for enhanced processing capabilities and improved performance compared to the traditional DTHNN. We will further discuss the details of this proposed model and its potential applications in the following sections.

3.1. Architecture-1:

$$\widetilde{V}(n+1) = Sign\,\{\acute{W}\,\widetilde{V}\,(n) - \widetilde{T}\} \quad (1)$$

with, $\widetilde{V}(0)$ as the initial state matrix. In (1), $\{\widetilde{V}(n)$: for $n \geq 0\}$ is a {+1, -1} valued state matrix and $\widetilde{T}$ is a matrix of thresholds (motivated by the idea in [9]). It should be noted that (1) corresponds to fully parallel mode of operation of 2-D AM. It readily follows that serial mode of operation of such an associative memory corresponds to updating just one component of $\widetilde{V}(n+1)$. We ensure that the diagonal elements of weight matrix W are all non-negative.

Based on the convergence theorem of ordinary HNN in the serial mode of operation, 2-D AM converges to a stable state (matrix of +1, -1) and to a cycle of length 2 in the fully parallel mode of operation. A more general model of AM motivated by the idea in [11] is the following one:

$$\widetilde{V}(n+1) = Sign\{(\acute{W}(n)\widetilde{V}(n) - \widetilde{T}\,(n)\}, \quad (2)$$

In the spirit of Parallel HAM in (1), we can use three/higher-dimensional state tensors. Thus, we have the following general multi-dimensional associative memory.

$$\widetilde{V}(n+1) = Sign\{\acute{W} \cdot \widetilde{V}(n) - \widetilde{T}\}, \quad (3)$$

Where $\{\widetilde{V}(n)$: for $n \geq 0\}$ are state tensors and '$\cdot$' denotes suitable inner product. Also, $\widetilde{T}$ is the tensor of thresholds.

Note: Synthesis of Hopfield Associative memory with desired stable states, when threshold vector is a zero vector was documented in [6] and when the threshold vector is a non-zero vector was documented in [15]. naturally generalises to tensor based linear operators with eigen values and eigen tensors. These results naturally apply for architecture-1 above.

4. Stacking of Parallel Hopfield Associative Memories

In the two-dimensional associative memory architecture proposed earlier, the synaptic weight matrix is assumed to remain the same during the updating of state vectors in parallel. However, to relax this assumption and explore alternative approaches, we introduce a modified architecture. This new architecture addresses the need for dynamic synaptic weight updates during the state vector updating process.

By incorporating this modification, we aim to improve the flexibility and adaptability of the associative memory model. The dynamic updating of synaptic weights opens up possibilities for more complex computational operations and enhanced memory retrieval capabilities. We will delve into the details of this revised architecture and discuss its potential advantages and applications in the subsequent sections.

4.1. Architecture-2:

We propose stacking based architecture where the input of 'M' HAMs/HNNs is a matrix of {+1, -1} i.e., Let $\widetilde{V}(0)$ be a {+1-1} component matrix. $\widetilde{V}(0) = [V_1(0)$ $V_2(0)$ ..... $V_M(0)]$

Note: In this architecture, a two dimensional {+1, -1} matrix is associated with a 2-D matrix of {+1, -1} whose columns correspond to the stable states of associative memories.

4.2. Architecture-3:

In the proposed architecture of the stack, each level consists of a distinct two-dimensional associative memory with its own unique initial state matrix. This arrangement allows for the convergence of each memory to a different stable state matrix.

By introducing multiple levels of associative memories with varying initial states, we aim to enhance the memory storage and retrieval capabilities of the overall system. Each level can store different patterns or information, and the convergence of each memory to a stable state represents the successful retrieval of the stored information.

This architecture provides a hierarchical and layered approach to associative memory, enabling the system to handle complex and multi-level associations between patterns or data. The convergence of each level to a distinct stable state matrix allows for parallel processing and retrieval of different sets of information.

Further details and discussions on the specific mechanisms and applications of this stacked architecture will be presented in subsequent sections, providing insights into its potential benefits and use cases.

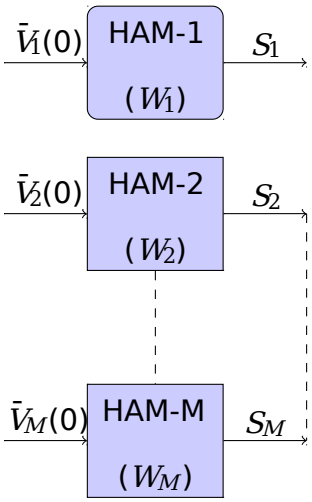

Figure 3.1: Block diagram of Architecture-2

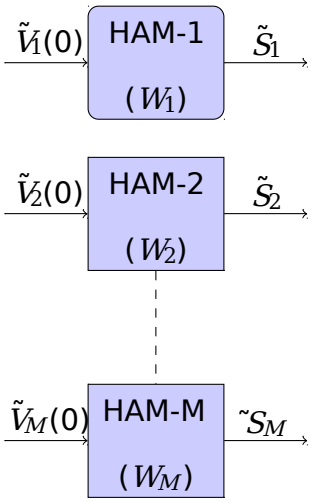

Figure 3.2: Block diagram of Architecture-3

Note: 3-Dimensional initial state tensor is associated with a 3-Dimensional stable state tensor.

It is possible to conceive architectures in which a higher dimensional information is associated with a lower dimensional information. We have following three cases:

- 2-Dimensional to 1-Dimensional
- 3-Dimensional to 2-Dimensional
- 3-Dimensional to 1-Dimensional

Such associative memories require higher dimensional stable states to be reduced to lower dimensional stable states.

CEILING NEURON - BASED ASSOCIATIVE MEMORY: 2-D to 1-D Associative Memory

In the Ceiling neuron model [9], each neuron is equipped with multiple thresholds. These thresholds are used to threshold the net contribution (i.e., $\sum_{i=1}^{N} W_i x_i$) of the neuron, resulting in a binary activation state. The state of the entire network in this model is represented as a matrix, where each element corresponds to the activation state of a neuron.

However, in Section-5 of [9], we propose an Associative Memory (AM) architecture where the state is represented as a vector. This means that the resulting stable states of the network are one-dimensional vectors. In this AM model, the higher-dimensional information is associated with lower-dimensional stable states.

The design of AMs that can associate higher-dimensional information with lower-dimensional states is an important research area. By leveraging techniques such

as thresholding and nonlinear transformations, it is possible to capture and represent complex patterns and associations in a compact and efficient manner.

The exploration of such AM architectures allows for the storage and retrieval of information in a compressed form, where higher-dimensional data can be efficiently represented and associated with lower-dimensional stable states. This can have practical applications in various domains, including image and pattern recognition, where high-dimensional data can be effectively processed and matched against stored representations.

Further details and advancements in the design and applications of AMs for associating higher-dimensional information with lower-dimensional states will be discussed in Section-5, providing insights into the capabilities and potential of this approach.

Note-1: When functions of human memory are understood, it becomes evident that higher dimensional information is “associated” with lower dimensional information (and vice-versa) in an effortless manner. One of our goals in this chapter is to arrive at models of ANN which can achieve these functions.

Note-2: Using Parallel, Stacked HNN architectures, 1-D/2-D/3-D information can be stored and retrieved. Thus, a total of Nine architectures [3 x 3] are possible and included two of them.

Note-3: With two stages of Stacked/ parallel architectures, there are twenty-seven (9 X 3) possible associative memories. The effort is to model, biological associative memories.

5. Novel Associative Memories: Deep Learning

In most of the applications 1-D/2-D/3-D data (vectors/matrices/3-D arrays) is corrupted by noise [10]. In the case of 1-D neural networks, the author proposed the concept of “HYBRID Neural Networks” [12]. In that research paper, AM (e.g., HNN) is utilized to filter the noise. The input vectors after filtering are fed to a Multi-Layer Perceptron, which performs classification. Generalizing the idea, we employ 2-D/3-D associative memory to filter noise from images, videos.

The filtered input is fed to a Deep Convolutional network for performing classification.

The Block diagram representation of such a Deep neural network is provided below.

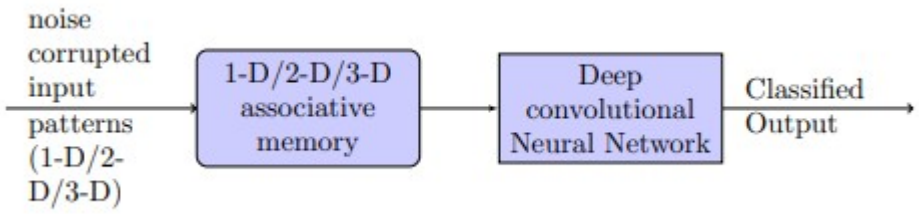

Figure 3.3: Block diagram of novel associative memories.

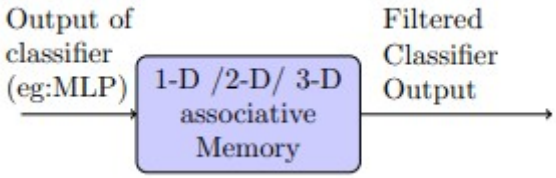

Figure 3.4: Block diagram of Filtered Classifier Output.

In the discussed associative memory architectures, the stable states play a crucial role in the input of the Convolutional Neural Network (CNN). These stable states can be synthesized to represent desired patterns, where each stable state corresponds to a specific class in a classification problem. By training the CNN using these desired stable states, the network can learn to perform the desired classification task.

In Fig. 4, an architecture is presented that combines the associative memory with the CNN, and it has been found to achieve good classification performance. The training process involves properly training the CNN to eliminate spurious stable states and focus on the desired stable states associated with the target classes.

It is important to note that the stable states can have different dimensionalities depending on the type of data being processed. For example, in audio/text-related data, the stable states may be one-dimensional, while in image or video data, they can be two-dimensional or three-dimensional.

The synthesis approach proposed in [15] allows for the programming of desired stable states, and it can be easily extended to work with 2-D or 3-D associative memories. This enables the design and training of associative memories that can handle complex patterns and associations in higher-dimensional data.

Furthermore, in [12], an architecture is introduced where the noisy output patterns of a Multi-Layer Perceptron (MLP) are filtered using an associative memory. This demonstrates the potential of combining different neural network models to improve performance and enhance noise suppression capabilities.

Overall, these approaches highlight the power and versatility of associative memories in handling different types of data and solving classification problems, while also addressing challenges such as noise suppression and stable state synthesis.

6. Applications

Indeed, the architectures proposed in the previous sections aim to model the memory mechanisms observed in humans, particularly in terms of their ability to store and retrieve 1-D, 2-D, and 3-D information through the process of association. These architectures are inspired by the convergence theorem-based Discrete Time Hopfield Neural Network (DTHNN), which has been successfully utilized to store 1-D patterns represented by +1's and -1's.

The goal is to extend this idea and innovate architectures that can capture the memory capabilities observed in humans. For example, given a speech input, the memory should be able to correlate it with the associated face image, representing the biological neural network's ability to store the relationship between speech signals and associated images.

In general, the human brain is capable of retrieving different types of information (1-D, 2-D, 3-D) by using association processes. The proposed architectures in the previous sections provide a glimpse into this capability by presenting two specific architectures out of the possible nine. These architectures are designed to model the human brain's ability to store and retrieve diverse types of information such as speech, images, and video signals.

It is also important to consider the network topology of interacting associative memories in the human brain. The brain likely consists of various associative memory units interconnected in specific ways, enabling the storage and retrieval of heterogeneous types of information. While the proposed architectures focus on specific aspects, the overall understanding of the brain's associative memory system involves considering the complex network topology and interactions between different memory units.

By developing and exploring these architectures, we can gain insights into the memory mechanisms of the human brain and potentially apply them to various applications in artificial intelligence and cognitive modelling.

7. Results of Implementation

We are currently investigating dedicated purpose hardware to implement 1-D /2-D /3-D associative memories. These memory units can potentially “Speed up” retrieval of stored 1-D/ 2- D/3-D information. We now provide some numerical results on data. We also provide some results related to implementation based on black and white images. The results and implementation of all the architectures mentioned in Section-III and Section-IV are provided below:

7.1. Architecture-1

We have taken 3x3 symmetric weight matrix, one state matrix having elements of +1, -1’s and one threshold matrix whose values are in the range of 0 to 1. We implemented AM in parallel mode of operation. Here we have taken, $W=\begin{bmatrix} 0 & -3 & -2 \\ -3 & 0 & -4 \\ -2 & -4 & 0 \end{bmatrix}$. As a result, it converges to a cycle of length 2.

7.2. Architecture-2

We have taken two HAMs having same or different symmetric 3x3 weight matrices and a state matrix of two different 3x3 state vectors are $\widetilde{V}(0)$ = [$\acute{V}_1(0)$ $\acute{V}_2(0)$]. We have taken, $W_1 ¿ \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix} \wedge W_2 ¿ \begin{bmatrix} 0 & 4 & 5 \\ 4 & 0 & 6 \\ 5 & 6 & 0 \end{bmatrix}$. Such an AM converged with cycle of length 2 in parallel mode of operation. Then we stack all the final state vectors, which leads to a matrix of 2-D.

7.3. Architecture-3

We have taken two HAMs having same or different symmetric 3x3 weight matrices and a state matrix of two different 3x3 state vectors as $\widetilde{V_1}(0)$ and $\widetilde{V_2}(0)$.

We have taken, $W_1 ¿ \begin{bmatrix} 0 & 4 & 5 \\ 4 & 0 & 6 \\ 5 & 6 & 0 \end{bmatrix} \wedge W_2 ¿ \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}$. Such AM converged with cycle of length 2 in parallel mode of operation. Then we stack all the final state matrices, which leads to 3-D. Finally, it is shown that 3-D input converges to 3-D output.

7.4. Implementation on Black and White Images:

Here we have taken a black & white image having size of 5x5. Then we converted it into a 5x5 matrix of elements {+1, -1}. Then it is given as input to HAM. It converges in parallel mode with cycle of length 2. Then we fix the weights.

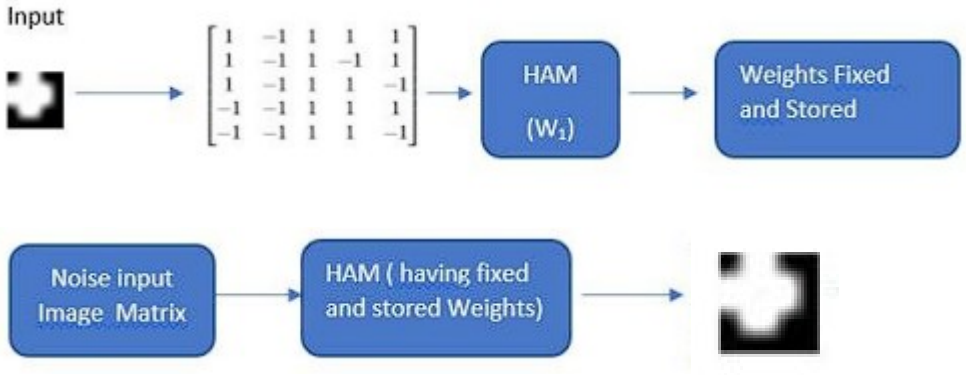

Figure 3.5: Block diagram of implementation on Black and White image

Now, add some noise to the input picture. Then apply this to the HAM having fixed weights. Finally, the original picture is reproduced. It is explained briefly in Fig.4.

8. Conclusions

In this chapter, the focus is on parallel and stacked associative memories, exploring their potential for storage and retrieval of 1-D, 2-D, and 3-D

information. The chapter presents various novel architectures of associative memories designed to handle different types of information.

The discussed architectures include the cascading of associative memories with other artificial neural network models, such as convolutional neural networks (CNNs). These architectures aim to leverage the strengths of associative memories in memory retrieval and combine them with the capabilities of CNNs for tasks such as pattern recognition and classification.

While some architectures have been proposed and discussed in this chapter, there are still six remaining architectures that are actively being investigated. These ongoing research efforts are focused on exploring the potential applications and utility of these architectures in practical scenarios.

By studying and developing these architectures, researchers aim to advance our understanding of associative memory systems and their capabilities for storing and retrieving various forms of information. The practical utility of these architectures lies in their potential to enhance artificial intelligence systems, cognitive modelling, and applications requiring memory retrieval and pattern recognition.

Overall, the chapter presents a glimpse into the current progress and ongoing research in the field of associative memories, highlighting the potential of these architectures for addressing complex information storage and retrieval challenges.

Chapter-4

On the Dynamics of Real-Valued Hopfield-Type Neural Networks based on Ceiling Neurons

1. Overview

In this chapter, we focus on the introduction of Hopfield neural networks based on the ceiling neuron model. Building upon the traditional binary Hopfield neural network, we extend its capabilities by considering a multistate set of possible states for the neurons. The motivation for this extension comes from the definition and properties of the ceiling neuron.

We examine the dynamics of the proposed model in both asynchronous and synchronous update modes. In the asynchronous update mode, we investigate the behaviour of the network when neurons are updated individually, one at a time. On the other hand, in the synchronous update mode, all neurons are updated simultaneously.

Through computational experiments, we demonstrate that the Hopfield neural network based on the ceiling neuron, under the usual conditions on synaptic weights (symmetric with non-negative diagonal elements), always converges to a stationary state in the asynchronous update mode. This means that the network settles down to a stable configuration where the neuron states no longer change.

In contrast, when considering the synchronous update mode, the Hopfield neural network with ceiling neurons can exhibit more complex dynamics. It is capable of generating limit cycles, similar to the behaviour observed in the classical Hopfield neural network. These limit cycles represent periodic patterns of neuron states that the network can exhibit over time.

Overall, this chapter presents the introduction of Hopfield neural networks based on the ceiling neuron model. We analyse their dynamics in asynchronous and synchronous update modes and showcase their ability to reach stable states or generate limit cycles, depending on the update scheme. The computational experiments provide insights into the behaviour and properties of these extended Hopfield neural networks.

2. Introduction

The Hopfield neural network (HNN) is one of the most important recurrent neural networks introduced in the literature. It was conceived by the American physicist W. A. Little in 1974 [22], and was popularized by the American physicist, biologist, and neurologist John Joseph Hopfield, in 1982 [13]. Hopfield investigated content addressable memories aimed to store and recall binary vectors using the Hebbian rule [11, 13]. Besides implementing content-addressable memories, HNNs have been applied in control [8, 29], classification [26, 35], computer vision and image processing [21, 34], and optimization [12, 20, 28].

Due to the limited representational capability of McCulloch-Pitt’s neurons, extensions of HNNs have been proposed aimed to process information of

different natures, especially information involving multidimensional data. In this sense, researchers have proposed, for example, complex-valued Hopfield neural networks, which allow to process 2-dimensional data as single entities by means of complex valued neurons [1, 2, 6, 9, 17, 24, 25, 30]. Furthermore, to process other kinds of multidimensional data as single entities, researchers have extended the HNNs from the field of real numbers to hypercomplex fields. Examples include quaternion valued HNNs [15, 16, 31, 32, 33], octonion-valued HNNs [5, 18, 19], and other hypercomplex-valued HNNs based on Cayley-Dickson or Clifford algebras, for instance [7].

Nevertheless, in this chapter we focus on another alternative, internally to the field of real numbers, to introduce a kind of neuron allowing to capture information of a multidimensional nature. The so-called ceiling neuron, introduced in [10], acts to increase the cardinality of the set of possible states of a real-valued neuron through the introduction of a finite set of thresholds to each neuron of an HNN. As we will see, this approach allows associating a multistate set of cardinalities equal to $K+1$ to each real-valued neuron of an HNN, where K is a positive integer called resolution factor.

This chapter is organized as follows: Section 3 introduces the ceiling neuron model and highlights its main operational differences compared to the classic McCulloch-Pitts neuron model. Sections 4 and 5 discuss aspects inherent to HNNs based on ceiling neurons and McCulloch-Pitt's neurons, with emphasis on their dynamics. In order to illustrate the dynamics of HNNs, Section 6 presents some computational experiments. We finish the chapter with some concluding remarks at Section 7.

3. The McCulloh-Pitt's and the Ceiling Neuron Model

The first computational model of a neuron was proposed by the neuroscientist Warren McCulloch and the logician Walter Pitts in 1943 [23]. Let us review this model. Consider a neuron called neuron i. Let real numbers $x_1, x_2, \ldots, x_N$ be the inputs of the neuron i, which, from a biological point of view, represent excitatory or inhibitory postsynaptic potentials at neural dendrites $1, 2, \ldots, N$. Associated with each of them we consider the real-valued synaptic weights $w_{i1}, w_{i2}, \ldots, w_{iN}$, respectively, which represent the intensities of the postsynaptic potentials at dendrites. The weighted sum $x_1 w_{i1} + x_2 w_{i2} + \ldots + x_N w_{iN}$ is called *action potential of the neuron i*, and is denoted in this chapter by v_i. Formally, we write

$$v_i = \sum_{j=1}^{N} w_{ij} x_j \quad (1)$$

The McCulloch–Pitt's neuron allows binary activations, i.e., it either "fires" with an activation equal to 1 or "does not fire" with an activation equal to 0. Specifically, if the action potential v_i is greater than a fixed threshold real value θ_i, the neuron i "fires". Otherwise, the neuron i "does not fire". This thresholding process can be mathematically represented by using the Heaviside-type step function h: $R \to \{0, 1\}$ defined by

$$h(x) = \begin{cases} 1, x > 0 \\ 0, otherwise \end{cases} \quad (2)$$

We obtain the output $y_i \in \{0,1\}$ of the neuron i by means of the equation

$$y_i = h(v_i - \theta_i) \quad (3)$$

Figure 1 illustrates the operating mode of a McCulloch-Pitts neuron.

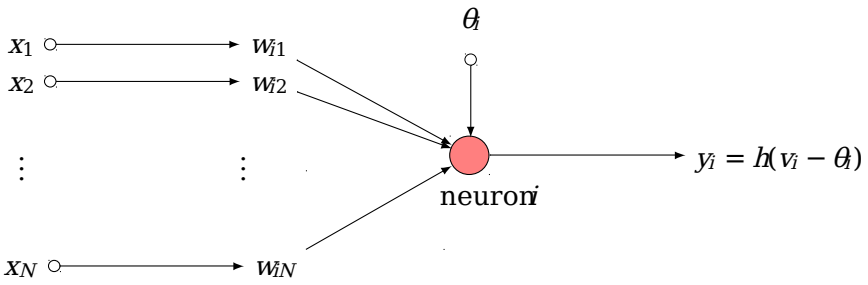

Figure 4.1: The McCulloch-Pitts neuron model.

Let us present the ceiling neuron [10]. Unlike McCulloch-Pitts neuron model, in the ceiling neuron model each neuron i has a set $\{\theta_{1i}, \theta_{2i}, \ldots, \theta_{Ki}\}$ of K distinct real-valued thresholds associated with it. Consequently, the set of possible values of a neuron is expanded from the binary set $\{0,1\}$ to the multistate set $S = \{0,1,\ldots,K\}$, where $K \geq 1$ is some positive integer. In the context of this chapter, the number K will be called *resolution factor*, due to its structural conformity with the resolution factors commonly used in complex-valued HNNs [1, 6, 9, 17, 24, 30].

Similar to the McCulloch-Pitts neuron model, consider a neuron i with real valued inputs $x_1, x_2, \ldots, x_N$, and the real-valued weights $w_{i1}, w_{i2}, \ldots, w_{iN}$, respectively. Besides, let us define the set of the real-valued thresholds associated with the neuron i by $\{\theta_{1i}, \theta_{2i}, \ldots, \theta_{Ki}\}$, with $\theta_{\mu i} = 6\ \theta_{\eta i}$ all $\mu = 6\ \eta$.

The output y_i of the ceiling neuron i is obtained as follows. For each threshold θ_{ji}, with $j = 1,\ldots,K$, we apply the Heaviside-type step function given by (2) on the difference $v_i - \theta_{ji}$, and add all the results obtained. Formally, the output of the ceiling neuron i is given by equation

$$y_i = \sum_{j=1}^{K} h(v_i - \theta_{ji}) \quad (4)$$

where v_i is the action potential of the neuron i given by (6). It is easy to see that $y_i \in S = \{0,1,...,K\}$.

Figure 2 illustrates the operating mode of a ceiling neuron.

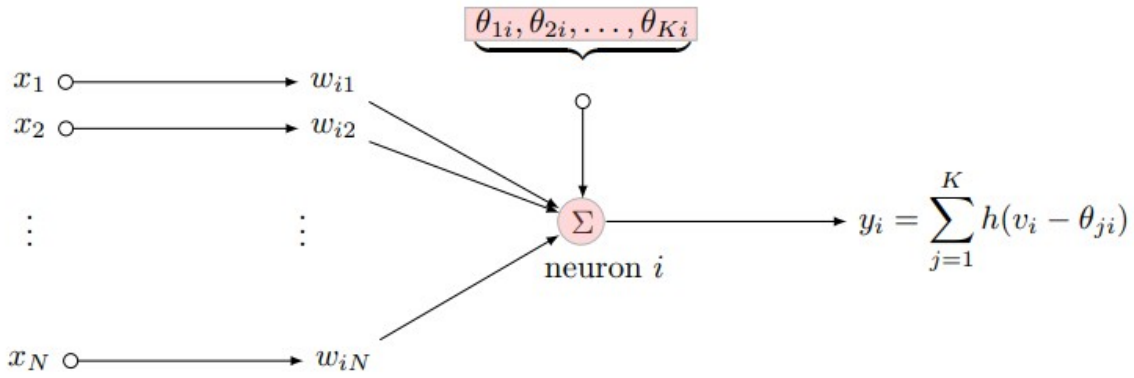

Figure 4.2: The ceiling neuron model

From a practical point of view, the output y_i of a ceiling neuron can be interpreted as the number of thresholds lower than the action potential v_i. In this sense, it is important to note that a single ceiling neuron divides the space of the values of a neuron into $K + 1$ subspaces of decision. In contrast, to mimic this same action, K neurons of McCulloch-Pitts would be needed.

HNNs based on ceiling neurons can be applied in reconstruction of grayscale or color images, image filtering, multiclass classification, semantic segmentation, or optimization, for instance, in a different way compared to a classic HNN approach, or a hypercomplex-valued HNN approach.

4. Binary Hopfield Neural Network

The binary HNN is one of the most important recurrent neural networks from the literature. It is composed by a totally connected single-layer with neurons of McCulloch-Pitts [13]. Consider a HNN with N neurons, let w_{ij} be the jth synaptic weight of the ith neuron, and let θ_i be the threshold of the ith neuron. The state of the ith neuron at time t is denoted by $x_i(t) \in \{0,1\}$, for $i = 1,...,N$. The output $x_i(t + 1)$ of the vector $\mathbf{x}(t + 1)$ can be obtained according to the equation

$$x_i(t+1) = \begin{cases} h\left(v_i(t) - \theta_i\right), v_i(t) - \theta_i \neq 0 \\ x_i(t), otherwise \end{cases} \quad (5)$$

where,

$$v_i(t)=\sum_{j=1}^{N} w_{ij}x_j(t)$$

(6)

is the *action potential of ith neuron at time t*, and h : $R \to \{0, 1\}$ is the Heaviside-type step function given by (2).

Note that Heaviside-type step function h : $R \to \{0, 1\}$ given by (2) is intrinsically related to the classic real sign function sgn : $R \to \{-1, 1\}$ by means of the invertible equation

$$\text{sgn}(x) = 2\text{h}(x) - 1$$

(7)

Equation (7) provides an easy way to convert an HNN with neuron values in {0, 1} (binary) to other HNN with values in {−1, 1} (bipolar), and vice versa. It is important to note that the dynamics of an HNN is invariant for this conversion.

The neurons of an HNN can be updated asynchronously or synchronously. If the neurons are updated asynchronously, that is, a single neuron is updated at time t, HNNs always settles at an equilibrium state if the synaptic weight satisfies the usual conditions $w_{ij}=w_{ji}$and $w_{ii} \geq 0$, that is, the synaptic weight matrix is symmetric with non-negative diagonal elements, for all i, j = 1, . . . , N, [4, 14]. This means that the sequences $\{x(t)\}_{t\geq 0}$ generated by the HNN are always convergent, given any initial state x(0) $\in \{0,1\}^N$. In an asynchronous update mode, the neurons can be updated randomly, or a pre-defined order can be imposed on the updates. In the computational experiments, we chose the following update order: first update the first neuron, then the second neuron, and so on.

In turn, in the synchronous update mode all neurons of the HNN are updated at the same time t. Some researchers see this mode of updating as less plausible from a biological point of view [27]. They base their assertions on the absence of scientific evidence about the existence of a global clock that influences or determines natural neural networks. Regardless of the biological plausibility, this mode of operation of an HNN occurs mathematically as follows.

Consider an HNN with N neurons, let $\mathbf{W}$ be a matrix such that w_{ij} be denotes the jth synaptic weight of the ith neuron, and θ a N-dimensional column vector with ith component θ_i equal to the threshold of the ith neuron. The vector $\mathbf{x}(t+1)$ is obtained from $\mathbf{W}$, θ, and $\mathbf{x}(t)$ according to the equation

$$x_i(t+1)=\begin{cases} h\left((Wx(t))_i-\theta_i\right), (Wx(t))_i-\theta_i\neq 0 \\ x_i(t), otherwise \end{cases} \quad (8)$$

where h is the Heaviside function given by (2).

By using the synchronous update mode, and imposing the same conditions on W, HNNs can produce limit cycles of length 2, that is, periodic sequences of period 2 can be generated by the equation (8). Further details on the dynamics of an HNN using synchronous update can be found in [3].

Remark 1: *It is important to highlight the difference between the equations (5) and (8). In equation (5), only the action potential of the ith neuron of x(t) is calculated at each time t by using the inner product between the ith row of* ***W*** *and the vector x(t). Then, the activation function h together with the threshold* θ_i *are used to update the vector x(t). The values of i and t are incremented, and the the process is repeated until some stopping criteria is satisfied. In turn, in the equation (8) all action potentials of the neurons 1, 2, . . . , N are calculated at the same time t using the usual matrix product between* ***W*** *and x(t). Then, the activation function h together with the components of the vector θ are used to update all neurons. After, the value of t is incremented, and the process is repeated until some stopping criteria is satisfied. Also note that, in general,* $(Wx(t))_i \neq v_i(t)$.

5. Real-valued Hopfield Neural Network based on Ceiling Neurons

In this section, we present an extension of the HNN model replacing the McCulloch Pitts neuron with the ceiling neuron. Consider a HNN with N neurons, and let $\mathbf{W}$ be the $N \times N$ matrix where w_{ij} denotes the jth synaptic weight of the ith neuron. Let us define the matrix with all thresholds of the HNN by $\mathbf{\Theta} \in R^{K\times N}$. Specifically, the ith column of $\mathbf{\Theta}$ contains the K distinct thresholds $\theta_{1i},\theta_{2i},...,\theta_{Ki}$ of the ith neuron, where $i \in \{1,2,...,N\}$. The state of the ith neuron at time t is denoted by $x_i(t) \in S = \{0,1,...,K\}$.

In an asynchronous update mode, each ceiling neuron i of the vector $\mathbf{x}(t+1)$ is obtained according to the equation

$$x_i(t+1)=\begin{cases}\sum_{j=1}^{K} h\left(v_i(t)-\theta_{ji}\right), v_i(t)-\theta_{ji}\neq 0,, \forall\, j\in\{1,2,...,K\} \\ x_i(t), otherwise\end{cases} \quad (9)$$

where $v_i(t)$ is given by (6), and h is the Heaviside-type step function given by (2).

Now, consider the HNN with N ceiling neurons in the synchronous update mode. Let $\mathbf{W} \in R^{N\times N}$ be a matrix of synaptic weights, and $\mathbf{\Theta} \in R^{K\times N}$ the matrix of thresholds of the HNN. In this case, the vector $\mathbf{x}(t+1)$ is obtained from $\mathbf{W}$, $\mathbf{\Theta}$, and $\mathbf{x}(t)$ according to the equation

$$x_i(t+1)=\begin{cases}\sum_{j=1}^{K} h\left(\left(Wx(t)\right)_i-\theta_{ji}\right), \left(Wx(t)\right)_i-\theta_{ji}\neq 0,, \forall\, j\in\{1,2,\ldots,K\} \\ x_i(t), otherwise\end{cases} \qquad (10)$$

where h is the Heaviside-type step function given by (2).

Remark 2: *Broadly speaking, the practical difference between the HNN with ceiling neurons given by the equation* (9)*, and the HNN given by* (10) *resides in the way in which the action potentials are obtained, accordingly to the remark 1, similarly to the classic HNNs.*

Theorem 1 presents sufficient conditions for an HNN with ceiling neurons always settles down at a stationary state.

Theorem 1: *Consider an HNN with N ceiling neurons and a resolution factor K ≥ 1. Let Θ* $\in R^{K\times N}$ *be the matrix of thresholds of the HNN, and S = {0, 1, . . ., K} the set of all possible values for a ceiling neuron. If the synaptic weights satisfy the conditions* $w_{ij}=w_{ji}$ and $w_{ii} \geq 0$ *for all i, j* $\in$ *{1, . . ., N }, then the sequences* $\{x(t)\}_{t\geq 0}$ *generated by the evolution equation (9) are always convergent given any initial state x(0)* $\in S^N$ *.*

Remark 3: *Theorem 1 is true for* $K=1$ *[12]. In this case, an HNN based on* N *ceiling neurons is equivalent, apart from an isomorphism, to an HNN based on the* N *McCulloch-Pitts neurons. The proof of the theorem follows, for example, from the introduction of the Lyapunov functional* E *:*

$\{0,1\}^N \rightarrow$ R *given by* $E(\mathbf{x}(t)) = \frac{-1}{2}x(t)^T W\, x(t)+\Theta x(t)$ *for the HNN [12].*

To prove Theorem 1 there is no need to build explicitly a Lyapunov functional for the HNN. Just note that the each vector $\mathbf{x}(t) \in S = \{0,1,\ldots,K\}^N$ can be decomposed as a sum of vectors in $\{0,1\}^N$. Precisely, let $M = \max\{\mathbf{x}(t)\} \leq K$ be the maximum value between all components of the vector x(t). Note that there exists a sequence of vectors $\mathbf{z}_\mu \in \{0,1\}^N$ such that $\sum_{\mu=1}^{M} z_\mu = x(t)$. Then, since HNNs with resolution factor $K=1$ satisfy

Theorem 1, HNNs with arbitrary K also satisfy. In other words, any vector obtained as an output from an HNN with $K > 1$ thresholds can be decomposed as the sum of a certain number of vectors obtained as outputs from an HNN with $K = 1$ threshold.

Remark 4: *If the synaptic weights satisfy the conditions* $w_{ji} = w_{ij}$ *and* $w_{ii} \geq 0$ *for all* $i,j \in \{1,...,N\}$*, then the sequences* $\{\mathbf{x}(t)\}_{t\geq 0}$ *generated by the evolution equation* (10) *are not always convergent given any initial state* $\mathbf{x}(0) \in S^N$.

The experiments from Section 5 illustrate what has been addressed so far.

6. Computational Experiments

Experiment 1: *The objective of this experiment is to compare, under Theorem 1 conditions, the dynamics of the asynchronous and synchronous HNN models with ceiling neurons*

For this, consider the symmetric synaptic weights matrix $\mathbf{W} = \begin{bmatrix} 0 & 3 & -2 \\ 3 & 0 & -4 \\ -2 & -4 & 0 \end{bmatrix}$, *the thresholds matrix* $\boldsymbol{\Theta} = \begin{bmatrix} 0.5 & 0.1 & 0.1 \\ 0.3 & 0.2 & 0.9 \end{bmatrix}$, *and the initial state* $x(0) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$ *for the HNN.*

1a) *In an asynchronous update, that is, by using the evolution equation (8), we obtain the following dynamics for the HNN.*

First iteration:

$$v_1(0) = 0 \cdot 0 + 3 \cdot 1 + (-2) \cdot 1 = 1.$$

Then, $x_1(1) = h(1 - 0.5) + h(1 - 0.3) = 1 + 1 = 2$. *Thus,*

$$x(1) = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}.$$

Second iteration:

$$v_2(1) = 3 \cdot 2 + 0 \cdot 1 - 4 \cdot 1 = 2.$$

Then, $x_2(2) = h(2 - 0.1) + h(2 - 0.2) = 1 + 1 = 2$. Consequently,

$$x(2) = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}.$$

Third iteration:

$$v_3(2) = -2 \cdot 2 - 4 \cdot 2 + 0 \cdot 1 = -12.$$

Then, $x_3(3) = h(-12 - 0.1) + h(-12 - 0.9) = 0 + 0 = 0$. Thus,

$$x(3) = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}.$$

Similarly, we obtain $x(3) = x(4) = \ldots = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$.

Therefore, the asynchronous version of the HNN settles down into the stationary state x(3) in 3 iterations.

1b) *In the synchronous update, that is, by using the evolution equation (10), the HNN produces a limit cycle of length 2. Indeed,*

At time t = 1:

$$\boldsymbol{W} \cdot x(0) = \begin{bmatrix} 0 & 3 & -2 \\ 3 & 0 & -4 \\ -2 & -4 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ -4 \end{bmatrix}.$$

Then,

$$x(1) = \begin{bmatrix} h(+1-0.5)+h(+1-0.3) \\ h(-4-0.1)+h(-4-0.2) \\ h(-4-0.1)+h(-4-0.2) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.$$

For t = 2:

$$\boldsymbol{W} \cdot x(1) = \begin{bmatrix} 0 & 3 & -2 \\ 3 & 0 & -4 \\ -2 & -4 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ -4 \end{bmatrix}.$$

Consequently,

$$x(2) = \begin{bmatrix} h(0-0.5)+h(0-0.3) \\ h(+6-0.1)+h(+6-0.2) \\ h(+6-0.1)+h(+6-0.2) \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}.$$

Finally, at time t = 3:

$$\boldsymbol{W} \cdot x(2) = \begin{bmatrix} 0 & 3 & -2 \\ 3 & 0 & -4 \\ -2 & -4 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \\ -8 \end{bmatrix}.$$

Then,

$$x(3) = \begin{bmatrix} h(+6-0.5)+h(+6-0.3) \\ h(0-0.1)+h(0-0.2) \\ h(-8-0.1)+h(-8-0.2) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = x(1).$$

In this case, we obtain the periodic sequence

$$\boldsymbol{x}(t)_{t\geq 0} = \{\boldsymbol{x}(0), \boldsymbol{x}(1), \boldsymbol{x}(2), \boldsymbol{x}(1), \boldsymbol{x}(2), \ldots\}.$$

Unlike the asynchronous HNN, the synchronous HNN with the same input parameters produced a limit cycle of length 2, similarly to the binary case.

Experiment 2: *This experiment aims to illustrate some differences between asynchronous and synchronous dynamics of an HNN with ceiling neurons.*

Consider the synaptic weights matrix $\boldsymbol{W} = \begin{bmatrix} 0 & -3 \\ -3 & 0 \end{bmatrix}$ *, and the thresholds matrix*

$\boldsymbol{\Theta} = \begin{bmatrix} -3.5 & -2.5 \\ 1.5 & -0.5 \end{bmatrix}$. *. In this case,* $S = \{0,1,2\}$ *because the resolution factor of the HNN is* $K = 2$*. The HNN has the following set of state vectors:*

$$V = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}$$

Representing the elements of V in the order in which they are arranged by the blue edges, marked with the digits 0, 1, . . . , 8 respectively, we can quickly visualize the dynamics generated by the Equation (9) (asynchronous update mode) through the directed graph illustrated by Figure 3. Note that the vector

represented by the edge 3, corresponding to $\begin{bmatrix}1\\0\end{bmatrix}$*, is the state vector that most attracts other state vectors. The exceptions are the state vectors represented by the edges 2, 5, and 8.*

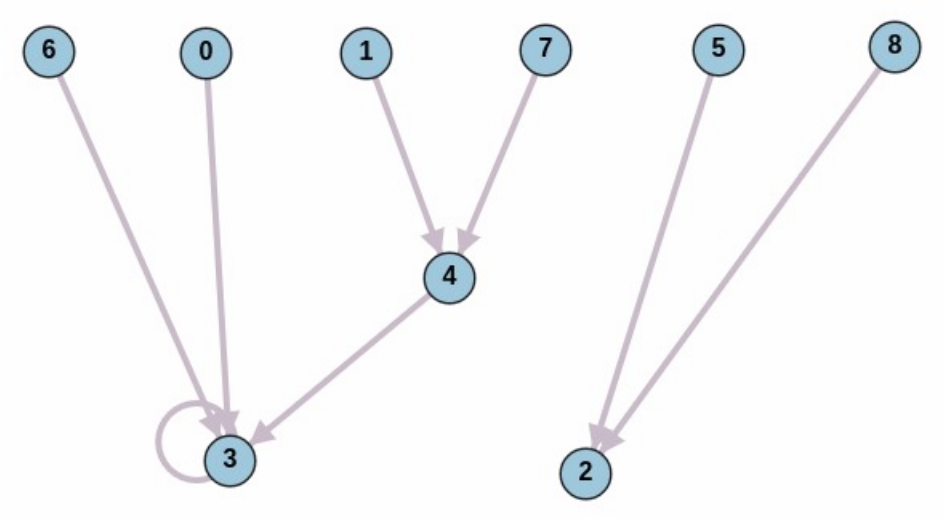

Figure 4.3: Directed graph illustrating the dynamics of the asynchronous HNN given by equation (9).

Similarly, consider the same matrices $\mathbf{W}$ *and* $\boldsymbol{\Theta}$*, but the synchronous HNN, according to the equation* (10)*. The dynamics of the HNN is represented by the*

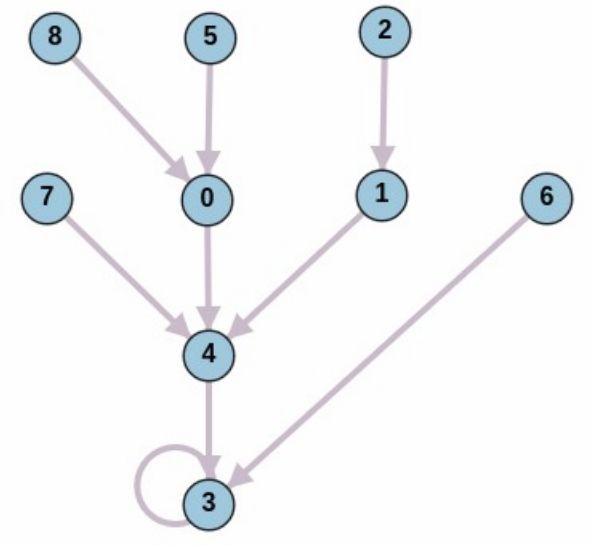

Figure 4.4: Directed graph illustrating the dynamics of the synchronous HNN given by equation (10).

directed graph of Figure 4. Note that the dynamics of relaxation of the HNN in the synchronous update is different from that of the asynchronous update, as we might expect. In the synchronous update, the state vector $\begin{bmatrix}1\\0\end{bmatrix}$ *is the only attractor of all the other vectors. In this particular experiment, by using asynchronous update mode, the number of stationary states in the HNN is double than the number of stationary states in synchronous updating. From the point of view of the use of HNNs for the implementation of content-addressable memories, this example suggests that the HNN with asynchronous updating could provide greater storage capacity than the synchronous HNN. Despite this, the HNN with the synchronous update mode does not always generate convergent sequences, which would hamper its use in the implementation of content-addressable memories. See the remark* 4*, and the Experiments* 1 *and* 3.

Experiment 3: *In order to evaluate the convergence of the sequences produced by the HNN models with ceiling neurons, we proceeded as follows: We first generated* 100×100 *real-valued matrix* $\boldsymbol{R}$ *with entries* r_{ij} *= randn, where randn yields a random scalar drawn from the standard normal distribution. Then, we computed* $\boldsymbol{U}=\frac{1}{2}\left(R+R^{T}\right)$*, where* $\boldsymbol{R}^{T}$ *denotes the transpose of* $\boldsymbol{R}$*, and defined the synaptic weight matrix by*

$$\boldsymbol{W} = \boldsymbol{U} - diag(u_{11},u_{22},...,u_{100100}),$$

where $diag(u_{11},u_{22},...,u_{100100})$ *is the diagonal matrix composed of the diagonal elements of* $\boldsymbol{U}$*. Note that* $\boldsymbol{W}$ *satisfies* $w_{ij} = w_{ji}$ *and* $w_{ii} = 0$ *for all any indexes* i *and* j*. Figure 5 shows the probability of a randomly generated HNN model with ceiling neurons settles down into an equilibrium state in at most* 1000 *iterations, that is, we allowed the neural networks to evolve while* $t \leq 1000$*. Success probabilities have been computed by repeating the procedure* 1000 *times for each resolution factor* $K \in \{1,2,...,10\}$*, and using the frequentist definition of probability to obtain a relative measure of the number of times that each HNN settled down to a stationary state. We would like to point out that the synaptic weight matrix* $\boldsymbol{W}$*, and the initial states* $\boldsymbol{x}(0)$ *were the same*

for all HNN models and were obtained by drawing each component of the set $\{0,1,\ldots,K\}^{100}$ using a uniform distribution. The elements of thresholds matrix $\boldsymbol{\Theta}$ were obtained by drawing each component from of the real interval $[-10,10]$ ensuring that each column of $\boldsymbol{\Theta}$ does not have repeated elements. Note that the probability of the synchronous model reaching a stationary state is always less than the probability of the corresponding asynchronous model. In addition, this probability decreases dramatically with the increase of the resolution factor K. In contrast, the probability of its asynchronous version to settle down into a stationary state is always equal to 1, which corroborates with Theorem 1.

7. Conclusion and Remarks

In this chapter, we explore the extensions of the Hopfield neural network by incorporating ceiling neurons. We begin by providing a theoretical overview of the classic Hopfield neural network model. Building upon the mathematical definition of a ceiling neuron, we introduce a modified Hopfield neural network that expands the set of possible states for each neuron beyond the binary values of {0, 1}. Instead, we define a multistate set S = {0, 1, ..., K}, where K is a positive integer referred to as the resolution factor.

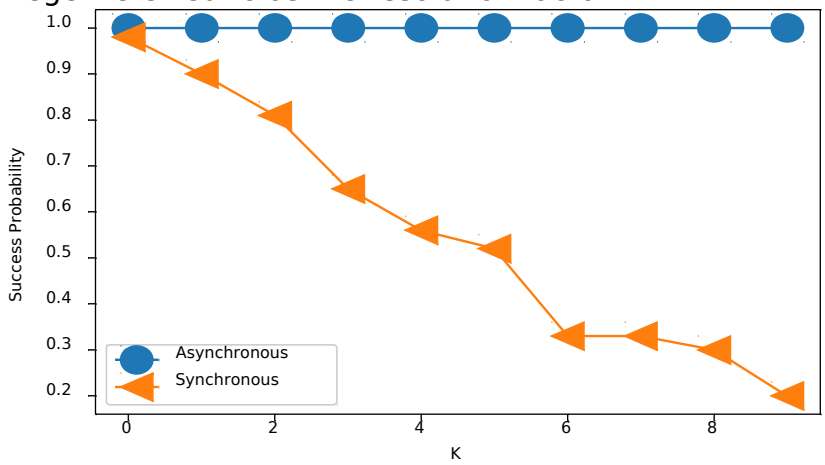

Figure 4.5: A comparison between the dynamics of HNN models with K ceiling neurons.

Next, we delve into the dynamics of the proposed models, considering both asynchronous and synchronous update modes. The asynchronous version of the model demonstrates a tendency to settle into a stationary state consistently, given that the synaptic matrix W exhibits symmetry with non-negative diagonal

elements. On the other hand, the synchronous model does not always generate convergent sequences under the same conditions on the W matrix.

To provide a practical perspective, we conduct computational experiments that illustrate the dynamics discussed throughout the chapter. These experiments help to further comprehend the behaviour of the proposed models in various scenarios.

As a future direction, we aim to implement content-addressable memories using the ceiling neuron-based models, employing different storage approaches. This will allow us to store and retrieve grayscale and colour images, providing an alternative solution to existing complex-valued and quaternion-valued HNN models. Specifically, we are interested in investigating the role of increasing the number of thresholds for each neuron, as it pertains to the possibility of utilizing the proposed models as specific image filters.

By exploring these research avenues, we anticipate gaining a deeper understanding of the capabilities and potential applications of the Hopfield neural network with ceiling neurons in image processing and memory retrieval tasks.

Chapter-5

Investigation of the proposed methods in this thesis

- ***Noise Suppression in Artificial neural networks (ANN's)***

In the context of SVM's, Vapnik attempted the problem of increasing NOISE MARGIN using support vectors for 2-class classification problem. To the best of our knowledge, there is no explicit investigation of Noise Suppression in Feed-forward networks such as Single/Multi-layer perceptron (MLP), CNN's. Motivated by Vapnik's work, we attempted and succeeded in increasing the robustness of ANN's.

In the next section, we discussed the concept of Null vectors of ANN's. We particularly reasoned that the convolution operation (by a mask/filter/kernel) in CNN's removes the frequency selective noise by Linear filtering. Also, with average pooling operation, another linear filtering stage is introduced.

In the case of Feed-forward neural networks such as Single Layer perceptron, there is a Linear transformation of input pattern vector by the synaptic weight matrix. Thus, noise vectors which lie in the null space of such matrix is suppressed. In MLP, locally from one layer to the next layer, the Linear transformation suppresses the noise vectors in its null space, but the sigmoidal nonlinearity acts on the locally filtered pattern vector. Thus, noise affecting the input patterns is suppressed at each stage.

But if the neurons in the hidden layer are Linear neurons, then multiple linear ($B_i's$) transformations (j of them) are cascaded i.e., effective linear transformation is

$$\acute{A}=\overline{B_1}\,\overline{B_2}\ldots.\overline{B_j}$$

Thus, the noise vectors in the null space of effective linear transformation $\acute{A}$ are filtered. With Non-linear neurons the above inference doesn't hold.

In the case of Extreme Learning Machine [49], there is a single hidden layer whose weights from input are randomly chosen. By suitably choosing the weights, noise vectors in input patterns are suppressed.

- ***1-D/2-D/3-D Hopfield Associative Memories (HAM's)***

To the best of our knowledge, 2-Dimensional/ 3-Dimensional associative memories based on Hopfield Neural network are successfully proposed by me. Also, detailed noise immunity study of such associative memories is investigated by us.

In case of Signal processing, Stack filters [48] are proposed. In our research, we proposed stacking of Hopfield associative memories to handle 2-D/3-D input signals.

This thesis brings to light many noise immunity issues in ANN’s. We expect research investigations by other researchers in future based on our efforts.

Chapter-6
Conclusion of Thesis

In conclusion, this thesis has made significant contributions to the field of noise suppression in Artificial Neural Networks (ANNs). By introducing the concept of null vectors, the thesis successfully addresses the issue of noise in perceptron-based models, providing a robust solution for noise suppression.

Furthermore, the thesis proposes the use of stacked parallel Hopfield neural networks to achieve noise suppression in various dimensions, including 1-D, 2-D, and 3-D associative memories. This approach offers a comprehensive framework for effectively dealing with noise and enhancing the reliability of information retrieval and pattern recognition tasks.

Additionally, the thesis presents a novel approach by proposing a real-valued Hopfield neural network based on the ceiling neurons model. Through associated experiments, the efficacy of this approach in noise suppression is demonstrated, highlighting its potential for application in deep neural networks.

The findings and outcomes of this thesis hold significant value for the field of noise suppression in deep neural networks. By providing effective techniques and models for mitigating noise interference, the thesis contributes to improving the overall performance and reliability of ANNs in real-world applications.

The knowledge and insights gained from this research can be leveraged to develop more robust and efficient neural network architectures that can effectively suppress noise and enhance the accuracy of data processing and decision-making. Furthermore, the techniques and models proposed in this thesis can serve as a foundation for future research in noise suppression and contribute to advancements in the broader field of artificial intelligence.

In conclusion, this thesis makes an important contribution to the field of noise suppression in Artificial Neural Networks by introducing novel techniques, models, and approaches. The results obtained through experimental evaluations provide evidence of the effectiveness of these methods, offering promising avenues for further research and practical implementation. Ultimately, the outcomes of this thesis are expected to have a significant impact on noise suppression in deep neural networks, improving their reliability and applicability in real-world scenarios.

Publications

- Noise suppression by Artificial Neural Network.
  AWICT 2017: Proceedings of the Second International Conference on Advanced Wireless Information, Data, and Communication Technologies
  November 2017
  https://doi.org/10.1145/3231830.3231847

- 1-D/2-D/3-D Hopfield Associative Memories.
  IOP Conference Series: Materials Science and Engineering, Volume 1049, International Conference on Artificial Intelligence and Machine Learning (ICAIML 2020) 4TH-5TH September 2020, Jaipur, India
  https://iopscience.iop.org/article/10.1088/1757-899X/1049/1/01200

- On the Dynamics of Real-Valued Hopfield-Type Neural Networks based on Ceiling Neurons
  Science Gate Publishing (SGP).

Bibliography

[1] AIZENBERG, I. *Complex-Valued Neural Networks with Multi-Valued Neurons*. Springer Publishing Company, Incorporated, 2016.

[2] AIZENBERG, N. N., AND AIZENBERG, I. N. 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* (1992), pp. 36–42.

[3] BRUCK, J. On the convergence properties of the Hopfield model. *Proceedings of the IEEE 78*, 10 (1990), 1579–1585.

[4] COHEN, M. A., AND GROSSBERG, S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*, 5 (1983), 815–826.

[5] DE CASTRO, F. Z., AND VALLE, M. E. 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.* (2018), vol. 06, pp. 2112–2118.

[6] DE CASTRO, F. Z., AND VALLE, M. E. Some remarks on the stability of discrete-time complex-valued multistate Hopfield neural networks. In *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics* (2018), vol. 6.

[7] DE CASTRO, F. Z., AND VALLE, M. E. A broad class of discrete-time hypercomplex-valued Hopfield neural networks. *Neural Networks 122* (2020), 54 – 67.

[8] GAN, J. Discrete Hopfield neural network approach for crane safety evaluation. In 2017 *International Conference on Mechanical, System and Control Engineering (ICMSC)* (May 2017), pp. 40–43.

[9] GARIMELLA, R. M. Some novel real/complex-valued neural network models. In *Computational Intelligence, Theory and Applications* (Berlin, Heidelberg, 2006), B. Reusch, Ed., Springer Berlin Heidelberg, pp. 473–483.

[10] GARIMELLA, R. M., MUNUGOTI, S. D., AND RAYALA, A. Novel ceiling neuron model and its applications. In 2019 *International Joint Conference on Neural Networks (IJCNN)* (2019), pp. 1–8.

[11] HEBB, D. *The Organization of Behavior*. John Wiley & Sons, New York, 1949.

[12] HOPFIELD, J., AND TANK, D. Neural computation of decisions in optimization problems. *Biological Cybernetics* 52 (1985), 141–152.

[13] HOPFIELD, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences 79* (Apr. 1982), 2554–2558.

[14] HOPFIELD, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences 81* (May 1984), 3088–3092.

[15] ISOKAWA, T., HISHIMURA, H., KAMIURA, N., AND MATSUI, N. Associative Memory in Quaternionic Hopfield Neural Network. *International Journal of Neural Systems 18*, 02 (2008), 135–145.

[16] ISOKAWA, T., NISHIMURA, H., AND MATSUI, N. Quaternionic Neural Networks for Associative Memories. In *Complex-Valued Neural Networks*, A. Hirose, Ed. Wiley-IEEE Press, 2013, pp. 103–131.

[17] JANKOWSKI, S., LOZOWSKI, A., AND ZURADA, J. Complex-Valued Multi-State Neural Associative Memory. *IEEE Transactions on Neural Networks 7* (1996), 1491–1496.

[18] KUROE, Y., AND IIMA, H. A model of Hopfield-type octonion neural networks and existing conditions of energy functions. In 2016 *International Joint Conference on Neural Networks (IJCNN)* (2016), pp. 4426–4430.

[19] KUROE, Y., IIMA, H., AND MAEDA, Y. Four models of Hopfield-type octonion neural networks and their existing conditions of energy functions. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), pp. 1–7.

[20] LI, C., YU, X., HUANG, T., CHEN, G., AND HE, X. A generalized Hopfield network for non-smooth constrained convex optimization: Lie derivative approach. *IEEE Transactions on Neural Networks and Learning Systems 27* (11 2015), 1–14.

[21] LI, J., LI, X., HUANG, B., AND ZHAO, L. Hopfield neural network approach for supervised nonlinear spectral unmixing. *IEEE Geoscience and Remote Sensing Letters 13*, 7 (July 2016), 1002–1006.

[22] LITTLE, W. A. The existence of persistent states in the brain. *Mathematical biosciences 19*, 1-2 (1974), 101–120.

[23] MCCULLOCH, W., AND PITTS, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics 5* (1943), 115–133.

[24] MÜEZZINOĞLU, M., GÜZELIŞ, C., AND ZURADA, J. A New Design Method for the Complex-Valued Multistate Hopfield Associative Memory*. IEEE Transactions on Neural Networks 14*, 4 (July 2003), 891–899.

[25] NOEST, A. J. Discrete-state phasor neural networks. *Physical Review A 38* (Aug 1988), 2196–2199.

[26] PAJARES, G., GUIJARRO, M., AND RIBEIRO, A. A Hopfield neural network for combining classifiers applied to textured images. *Neural Networks 23*, 1 (Jan. 2010), 144–153.

[27] PARISI, G. I., KEMKER, R., PART, J. L., KANAN, C., AND WERMTER, S. Continual lifelong learning with neural networks: A review. *Neural Networks 113* (2019), 54–71.

[28] SERPEN, G. Hopfield Network as Static Optimizer: Learning the Weights and Eliminating the Guesswork. *Neural Processing Letters 27*, 1 (2008), 1– 15.

[29] SONG, Y., XING, B., GUO, L., AND XU, X. System parameter identification experiment based on Hopfield neural network for self-balancing vehicle. In 2017 *36th Chinese Control Conference (CCC)* (July 2017), pp. 6887–6890.

[30] TANAKA, G., AND AIHARA, K. Complex-Valued Multistate Associative Memory with Nonlinear Multilevel Functions for Gray-Level Image Reconstruction. *IEEE Transactions on Neural Networks 20*, 9 (Sept. 2009), 1463– 1473.

[31] VALLE, M. E. An Introduction to Complex-Valued Recurrent Correlation Neural Networks. In *Proceedings of the IEEE World Conference on Computational Intelligence 2014 (WCCI 2014)* (Beijing, China, July 2014).

[32] VALLE, M. E., AND DE CASTRO, F. Z. Theoretical and computational aspects of quaternionic multivalued Hopfield neural networks. In 2016 *International Joint Conference on Neural Networks (IJCNN)* (July 2016), pp. 4418– 4425.

[33] VALLE, M. E., AND DE CASTRO, F. Z. On the dynamics of Hopfield neural networks on unit quaternions. *IEEE transactions on neural networks and learning systems 29*, 6 (2017), 2464–2471.

[34] WANG, Q., SHI, W., ATKINSON, P. M., AND LI, Z. 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 (March 2015), 1339–1352.

[35] ZHANG, H., HOU, Y., ZHAO, J., WANG, L., XI, T., AND LI, Y. 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* (2017), 1035 – 1043.

[36] J. BRUCK AND M. BLAUM, Neural Networks, Error Correcting codes and polynomials over the binary cube, *IEEE transactions on Information Theory*, vol.35, no.5, September 1989

[37] G. RAMA MURTHY, Optimal Robust Filter Models of Synapse: Associated Neural Networks, *Proceedings of International Conference on Soft Computing and Intelligent Systems*, December 27-29, 2007.

[38] https://math.stackexchange.com/questions/786108/explicit-formula-for-inverse-of-upper-triangular-toeplitz-matrix-inverse.

[39] MURTHY, G. RAMA. “Multi-Dimensional Neural Networks: Unified Theory”. *New Age International*, 2008

[40] GARIMELLA RAMA MURTHY, MONCEF GABBOUJ, “On the design of Hopfield Neural Networks: Synthesis of Hopfield type associative memories,” *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2015)*, July 2015, appears on IEEE Explore.

[41] H. SUN, H. HASEGAWA, I. YAMADA, “Multidimensional associative memory neural network to recall nearest pattern from input [image matching example],” 2005.

[42] G. RAMA MURTHY, “Multi / Infinite Dimensional Neural Networks, Multi/ Infinite Dimensional Logic Theory,” Published in 1995 *IEEE International*

Symposium on Circuits and Systems, Seattle. International Journal of Neural Systems, Vol.14. No.3, pp.223-225,2005...Generalization of Boolean Logic Theory.

[43] G. RAMA MURTHY, M. DILEEP, R. ANIL, “Convolutional Associative Memory*,” International Conference on Neural Information Processing [ICONIP 2015]*, November 2015, Turkey.

[44] G. RAMA MURTHY, L. BEHRA, “Adaptive Associative Memory,” Published in *GESTS International Transactions on Communications and Signal Processing*, April 2006. Also, in *Proceedings of National Conference on Soft Computing, Bhubaneswar*, 24th- 26th March-2006.

[45] G. RAMA MURTHY, “Hybrid Neural Networks,” *Proceedings of International Conference on Power System Analysis, Control and Optimization (PSACO-2008)*, 13th- 15th March-2008.

[46] IGOR AIZENBERG, “Complex-Valued Neural Networks with Multi-Valued Neurons,” Proceedings of International Conference on Power System Analysis,2011.

[47] RAMA MURTHY GARIMELLA, KRISHNA VAMSHI REDDY LANKALA, DEVAKI NIMMAGADDA, SREE DIVYA BONDALAPATI, “Synthesis/Programming of Hopfield Associative Memory,” ICMLDS-2019, ACM Digital Library, December 2019.

[48] P. Wendt, E. Coyle and N. Gallagher, "Stack filters," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 34, no. 4, pp. 898-911, August 1986, doi: 10.1109/TASSP.1986.1164871.

[49] Guang-Bin Huang, Qin-Yu Zhu and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), Budapest, Hungary, 2004, pp. 985-990 vol.2, doi: 10.1109/IJCNN.2004.1380068.

[50] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015). https://doi.org/10.1038/nature14539

[51] Haykin, S., 1994. *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

[52] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.