HIERARCHICAL ASSOCIATIVE MEMORY BASED ON OSCILLATORY NERUAL NETWORK

by

Yan Fang

Bachelor of Science, Xidian University, P. R. China, 2010

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2013

UNIVERSITY OF PITTSBURGH

SAWNSON SCHOOL OF ENGINEERING

This thesis was presented

by

Yan Fang

It was defended on

April 2nd, 2013

and approved by

Dr. Steven P. Levitan, PhD, Professor, Department of Electrical and Computer Engineering

Dr. Yiran Chen, PhD, Assistant Professor, Department of Electrical and Computer Engineering

Dr. Kartik Mohanram, PhD, Assistant Professor, Department of Electrical and Computer Engineering

Dr. Donald M. Chiarulli, PhD, Professor, Department of Computer Science

Thesis Advisor: Dr. Steven P. Levitan, Professor,

Electrical and Computer Engineering Department

HIERARCHICAL ASSOCIATIVE MEMORY BASED ON

OSCILLATORY NERUAL NETWORK

Yan Fang, M.S

University of Pittsburgh, 2013

In this thesis we explore algorithms and develop architectures based on emerging nano-device technologies for cognitive computing tasks such as recognition, classification, and vision. In particular we focus on pattern matching in high dimensional vector spaces to address the nearest neighbor search problem. Recent progress in nanotechnology provides us novel nano-devices with special nonlinear response characteristics that fit cognitive tasks better than general purpose computing. We build an associative memory (AM) by weakly coupling nano-oscillators as an oscillatory neural network and design a hierarchical tree structure to organize groups of AM units. For hierarchical recognition, we first examine an architecture where image patterns are partitioned into different receptive fields and processed by individual AM units in lower levels, and then abstracted using sparse coding techniques for recognition at higher levels. A second tree structure model is developed as a more scalable AM architecture for large data sets. In this model, patterns are classified by hierarchical k-means clustering and organized in hierarchical clusters. Then the recognition process is done by comparison between the input patterns and centroids identified in the clustering process. The tree is explored in a "depth-only" manner until the closest image pattern is output. We also extend this search technique to incorporate a branch-and-bound algorithm. The models and corresponding algorithms are tested on two standard face recognition data-sets. We show that the depth-only hierarchical model is very data-set dependent and performs with 97% or 67% recognition when compared to a single large associative

memory, while the branch and bound search increases time by only a factor of two compared to the depth-only search.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Steven P. Levitan, for his constant support and unlimited patience. I would like to thank Dr. Donald M. Chiarulli, Dr. Yiran Chen and Dr. Kartik Mohanram for their support and help in my thesis defence. Also, I extend my gratitude to Joni, Sasha and Sandy. This work could not have been done without their help. I especially would like to thank Dr. Anna C. Balazs for her desserts.

A lot of thanks to Xiang, Chet, Sam and all my dear friends for sharing both happy and difficult time with me. Thanks to Nan, my beloved girl who has the same enthusiasm for research. Finally, thanks to my parents, Jian Fang and Qiwen Yan, for their endless love and instruction.

I want to give my love to all of you, to the scientific research that drives our brilliant civilization forward, and to the vast and wonderful world that gave us life and keeps us guessing.

"I was like a boy playing on the sea-shore, and diverting myself now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me."

By Sir Isaac Newton

1.0 INTRODUCTION

The goal of this thesis is to explore algorithms and to develop architectures based on emerging nano-device technologies for cognitive computing tasks such as recognition, classification and vision. Though the performance of computing systems have been continuously improving for decades, for cognitive tasks that humans are adept at, our machines are usually ineffective despite the rapid increases in computing speed and memory size. Moreover, given the fact that CMOS technology is approaching the limits of scaling, computing systems based on traditional Boolean logic may not be the best choice for these special computing tasks due to their massive parallelism and complex nonlinearities. On the other hand, recent research progress in nanotechnology provides us opportunities to access alternative devices with special nonlinear response characteristics that fit cognitive tasks better than general purpose computation. Building circuits, architectures, and algorithms around these emerging devices may lead us to a more efficient solution to cognitive computing tasks or even artificial intelligence. This thesis presents several non-Boolean architecture models and algorithms based on nano-oscillators to address image pattern recognition problems.

1.1 MOTIVATION

With the increasing development of integrated circuit technology, computers have become increasingly more powerful due to the dramatic improvement in processor performance and memory capacity. However, the basic structure of our computing system remains almost unchanged since the invention of the first electronic computer in the 1940s [1]. Most modern computers still follow the traditional stored-program structure and use either Von Neumann or Harvard architectures [2][3]. These machines, based on Boolean logic, can execute the instructions stored in memory and perform arithmetic or logic computing at very high speed. Nonetheless, the cognitive tasks that are easy for human beings are usually very difficult for our current computing systems. This is because computers, which greatly differ from human brains, cannot efficiently associate, abstract and infer conclusions from input information by processing fuzzy and non-Boolean data. For decades, the development of artificial intelligence and neural networks has focused on the abstraction of models for algorithm design, while neglecting the fact that these cognitive functions and models do not fit the current computer architecture. For example, the neural network in a human brain processes large chaotic information in parallel through a huge number of neurons and synapses. We can neither build up a similar network on chip with neuron numbers of such large scale, nor simply reproduce the nonlinearity of signal processing on synapses. A simple additive neural network model implemented in software costs too much time for training as the number of neurons increases [4]. Because the nonlinear action function of one neuron has to be modeled with some complex equations and computed on a Boolean logic system, and this process has to be repeated millions of times even for a small part of a brain neural network model. Therefore, given the high difference in information processing

mechanisms between computers and the human brain, it is difficult to develop self-directed abilities like inference and generalization on a computer.

People may argue that we can still model the cognitive process in human brain with faster and more powerful computers in the future. Unfortunately, the limit of the CMOS downscaling trend will impede the performance improvement of computers in the near future. For integrated circuits, a decrease in the transistor's dimension means higher packing density, higher transition speed and lower power dissipation [5]. Over past three decades, the constant downscaling of transistor dimensions based on CMOS VLSI technology has played a key role in the progress of computing systems that has provided us with higher performance, lower energy cost and smaller physical size. But when the transistor's dimension scales down, the gate-oxide thickness and voltage level should decrease as well. Lower voltage leads to higher source-to-drain leakage current stemming from the thermal diffusion of electrons [5]. In addition, quantum-mechanical tunneling will increase gate leakage currents as the gate-oxide thickness reduces [6]. Thus, it is difficult to find a solution to our problem from computer systems based on traditional CMOS technology.

Motivated by the scaling limit of traditional CMOS devices, researchers began to pursue novel alternative devices at the nano-scale. Some of these devices have special nonlinear responses in the time domain or the frequency domain that are quite different from the bi-stable step function of traditional CMOS devices. For example, some of them have more than two stable states, periodic response functions or sigmoid response functions, which can be mapped to similar natural dynamic systems such as human neural circuitry [7]. Spin torque oscillators are a type of frequency-coherent spin devices, based on the interaction of a spin-polarized current with a thin magnetic layer [8]. This device was discovered to have an oscillatory behavior based on

exchange coupling, which is similar to the weakly coupled neural network model [9]. Another popular nano-device, the memristor, has been called the missing non-linear passive two-terminal electrical component relating electric charge and magnetic flux linkage [10]. The conductance of a memristor is controlled by the flux across or charge through it, just like the potentiation of a biological synapse is adapted by ionic transmissions. Thus, a nanoscale memristor also has the potential to reproduce the behavior of a neuron synapse. Beside the two nano-devices mentioned above, there are many other new devices like resonant body transistors [11] and single-electron transistors [12]. These devices can provide opportunities for building novel architectures for cognitive tasks like pattern recognition or computer vision.

The idea of special hardware architectures for cognitive tasks is not new. Equivalent circuits for a neuron model go back at least to 1907, where the model was just simply composed of a resistor and a capacitor [13]. Since then, various models were implemented based on different technologies. In the late 1980's Carver Mead proposed the term neuromorphic system in reference to all the electronic implementations of neural systems [14]. Meanwhile, with the development of digital computing systems, theoretical and computational models for neural networks and the human brain have been improved and refined many times. From the McCulloch-Pitts neuron model to the Hodgkin and Huxley's model [15][16], computational neuroscience gives us precise mathematic descriptions of the neuron. However, models for a single neuron or synapse have not proved to be good enough to explain the cognitive ability of a whole brain. More recent research trends have focused on the hierarchical structure of human brain and vision system, like HTM [17], HMAX [18], etc. With the new opportunities from emerging nano-devices and neuroscience, we can take the advantage from both sides and design

new architectures for cognitive computing tasks based on non-Boolean functions from a more macroscopic view of a hierarchical model of the human brain, instead of single neural network.

1.2 WORK OVERVIEW

In our work, we build upon this background to take advantage of the unique processing capabilities of weakly coupled non-linear oscillators to perform the pattern recognition task. In particular, we focus on pattern matching in high dimensional vector spaces to solve the "k-nearest neighbor" search problem. We choose this problem due to its broad applicability to many machine learning and pattern recognition applications. Taking the image recognition problem as an example, most algorithms extract specific features from the original images or apply various transfers to them in the first step. No matter what the preprocessing algorithms are, finally feature vectors generated by these algorithms need to be compared with some memorized vectors and match the “nearest neighbor” in the same space for recognition.

This process is similar to the human’s cognitive action. In a general recognition process, we first preprocess information from our sensory organ and then associate it with abstract information in our memory. The associative process is a comparison operation based on quite different metric from the norms like Euclidean distance that we commonly use. In an associative memory, the mapping from input information to output information is stored so that the output information can be retrieved when an input is given. From the view of a dynamical system, information is stored at those minima of a system’s energy function. The representation of input information is the initial state, and the system finally converges to a “nearest” energy minimum

as the output. Thus, the distance metric of comparison is determined by the energy function of the system. This has been called a neurodynamical system.

It has been shown that weakly coupled nonlinear oscillator networks are capable of behaving as an associative memory [19]. In an oscillatory neural network model, the equilibria of an associative memory are replaced by synchronized oscillatory states with certain phases. In work by our group and others, we try to implement such an associative memory model with new nano-devices and peripheral circuits. Due to the high speed and high packing density of some devices, we can perform the "k-nearest neighbor" search problem in parallel with very high speed. Patterns or vectors can be represented by the phase or frequency of each oscillator. Patterns can be stored in the network by changing the coupling coefficients of these oscillators, with different technologies. During the recognition process, the matched pattern represented by phase or frequency is read out by the circuit's support devices.

In the work with a standard associative memory (AM) unit like this, we can deal with patterns in high dimensions vector spaces by partitioning them into segments and storing them in multiple AM units. We also design a hierarchical structure to organize these AM units. Image patterns are separated into different receptive fields and processed by individual AM units at the low level, and then abstracted at higher levels of the model. Another tree structure model for the AM architecture can be used for large data sets. In this model, AM units are nodes of a tree, and patterns are classified by hierarchical k-means clustering algorithms and stored in the leaf nodes of tree, while other nodes only memorize centroids of clusters. During the recognition process, input patterns are compared with cluster centroids and classified to sub-clusters iteratively till the correct pattern is retrieved.

1.3 PROBLEM STATEMENT

The main problem that thesis addresses, is to build a non-Boolean associative memory (AM) system for pattern recognition based on nano-oscillators. The detailed questions we address are the following:

- How to pick a model that fits the oscillator network? Different models use different coding methods to represent information. For example, the Palm model uses a binary sparse representation, which is quite different from the patterns represented by phases and frequencies in an oscillatory neural network. Another question is, should we use real numbers or a binary code?
- How to implement the function of associative memory on the hardware of nano-oscillators? In this thesis we do not consider the circuit design for the nano-devices, but it is necessary to match the oscillator network model to our mathematic model. One question is what kind of distance metric we should use to represent the behavior of oscillator network.
- Then we need determine the structure of AM unit. The pattern can be represented by phase or frequency. Many parameters trade off each other in the design, such as the capacity, retrieval accuracy, and retrieval speed of the main system. The design is also limited by the density problem of hardware. Given the difficulty of coupling too many oscillators, the size and capacity of the AM cannot be infinite. Thus a single AM unit is not able to store many patterns or patterns with high dimension (long vectors).
- We may design rational architectures to organize the AM units hierarchically. However, this may sacrifice retrieval performance compared to an ideal single AM. The

disadvantages of this architecture need to be analyzed and solved. If we cannot improve the model from all aspects, the solution may be to design another possible model.

- From the algorithm level, the main problem is how we group and store the data in the hierarchical model, which influences the retrieval performance. The data sets we can use for testing and simulation of our model also deserve consideration.

1.4 WORK PLAN

To address the problems above, in this thesis we perform the following work:

- We analyze three AM models, the Hopfield Network, the Palm Network, and the Oscillatory Neural Network, compare them to each other, and simulate the Oscillatory Neural Network based on a phase locked loop (PLL). By doing this, we demonstrate how to build a basic AM unit with oscillatory devices and determine to use a unary coded Palm network as the mathematic model of our AM unit.
- We propose a hierarchical AM model to solve the problem resulting from the hardware limitations. This model is analyzed in detail and simulated.
- We compare this model to a single fully connected flat Palm Model with the same function, and improve the performance of our model by adding information abstraction between levels. The complete model is tested with a set of binary images with different noise tests.
- We explore the possibility of dynamic online learning for the AM model.
- To deal with problems in the real number field and high dimension spaces, we design a second hierarchical model called the N-Tree Model. First, we use three different distance

metrics to model the behavior of the AM unit on real nano-oscillators. Then we use a hierarchical k-means clustering algorithm as the method for storing data for our tree structure model.

- For simulation, two standard image data set are introduced, ATT and FERET, containing gray-scale images. After simulation and analysis of the N-tree model, we add Branch and Bound search algorithms to improve the model's performance.

1.5 THESIS ORGANIZATION

In Chapter 2, we explore the previous work on associative memory. In particular we examine the Hopfield Network, the Palm model, and the weakly coupled oscillator neural network. Based on this work, we develop a model for the intrinsic mechanisms of associative memory, and find theoretical support for oscillator based associative memory. Experiments and simulations are explored for these models. We also discuss the oscillator neural network implemented on real nano-devices.

In Chapter 3, we propose a hierarchical model to solve the scaling problem of real nano-oscillator based associative memory. We discuss the AM design tradeoffs between capacity and performance. We then simulate our model on a multiple binary image recognition problem and analyze the performance. Moreover, we explore more possible functions for the hierarchical AM model such as dynamical learning and Bayesian inference.

In Chapter 4, we propose another AM architecture, the N-tree model, to solve the capacity problem. This model organizes AM unit into a tree structure and classifies data though hierarchical k-means algorithm. We use two standard grayscale image data sets to verify our

model and enhance our system from the algorithmic level by introducing branch and bound search.

Chapter 5 summarizes our work and the contributions of this thesis, draws conclusions and discusses future work.

2.0 BACKGROUND

This chapter reviews some of the selected work that inspired this project. Some concepts concerning associative memory and history are introduced. We analyze and simulate different AM models in detail and describe our idea for basic AM unit.

2.1 ASSOCIATIVE MEMORY

An associative memory is a brain like distributed memory that learns by association. Association has been known to be a prominent feature of human memory since Aristotle, and all models of cognition use association in one form or another as the basic operation [4]. Association takes one of two forms: auto-association and hetero-association [4]. Auto-associative memory is configured to store a set of patterns (vectors) and is required to be able to retrieve the original specific pattern when presented with a distorted or noisy version. In other words, when the input pattern is different from all the memorized patterns, the associative process of the AM retrieves a best match to the input pattern given some distance metric, like the Euclidian distance. Thus, the association is a nearest neighbor searching procedure.

As the example given in Figure 2.1, consider an auto associative neural network that has memorized three binary image patterns for the numbers ‘0’, ‘1’, and ‘2’. When a distorted image

‘1’ is fed into the network, the output pattern should be the stored image ‘1’, which is the closest pattern in this associative memory.

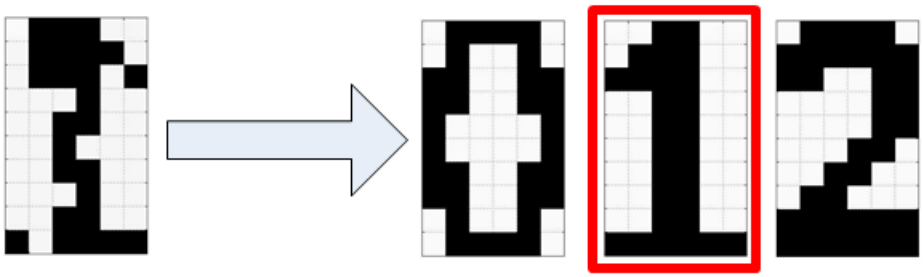


Figure 2.1 Example of Image Pattern Retrieval

Hetero-associative memory is the same except its output patterns are coded differently from the input patterns. Therefore we can consider that a mapping between input and output patterns is stored in the network for pattern retrieval. Hetero-association happens in many cognitive procedures. For instance, a group of neurons in one region of the brain activates neurons in multiple cortical regions.

In its simplest form, associative memory operations involve two phases: storage and recall. In the storage phase, we train the network with patterns by following a particular rule (for example, a Hebbian learning rule, or a Palm rule) that can form a weight matrix which indicates the weights of connectivity between neurons. Therefore in the training stage, the network stores the patterns we need in the interconnection weights. In the recall phase, the network performs pattern retrieval by using the interconnection weights to process a possibly noisy version of one of the stored patterns. It is assumed that the input pattern is based on one of the stored patterns.

The number of patterns stored in the associative memory provides a direct measure of the storage capacity of a network. In the design of associative memory, the tradeoff between storage capacity and correctness of retrieval is a challenge [4].

The earliest AM model was explored by Steinbuch and Piske in the 1960s [20], called Die Lernmatrix. It represents patterns with binary vectors and uses the Hebbian learning rule. Based on this model, Willshaw's group improved its information capacity with sparsely encoded binary vectors and proposed a model called an "Associative Net" [21][22]. They also tried to explain some brain network functions such as short term memory with their model [23]. In the 1980s Palm proved the storage capacity of Willshaw's model by a different method. Palm improved his model by introducing real-value weights and an iterative structure. Brain-state-in-a-box (BSB) is another AM model proposed by Anderson's group [24][25]. In 2007 the "Ersatz Brain Project" utilized BSB in building parallel, brain like computers in both hardware and software [26]. The Hopfield network invented by John Hopfield in the 1980s included the idea of an energy function in the computation of the recurrent networks with symmetric synaptic connections [27]. The most important contribution of the Hopfield Network is that it connects neural networks with physics, and the method of analyzing network stability from the viewpoint of a dynamic system. There also exist many other AM models which are not included in this brief review. Rather, we detail three important ones in this chapter, which are the inspiration of our work.

2.2 PALM NETWORK

A simple and effective associative memory model was proposed by Willshaw and Palm [21][28]. The hetero-associative Palm model is an additive neural network which contains two-layers of fully connected neurons, shown in Figure 2.2. Both the input pattern $\boldsymbol{x}$ and the output pattern $\boldsymbol{y}$ are binary coded vectors of n and m respectively. Assume we have

$$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \tag{2-1}$$

$$\boldsymbol{y} = [y_1, y_2, \cdots, y_m]^T \tag{2-2}$$

The Palm model performs the mapping of

$$\boldsymbol{x^k} \rightarrow \boldsymbol{y^k}, k = 1,2, \cdots, q \tag{2-3}$$

For some number of patterns q

The number of neurons and the size of input and output layers are determined by the vectors' length n and m, while q is the number of patterns. When x and y are identical, the Palm model becomes an auto-associative memory.

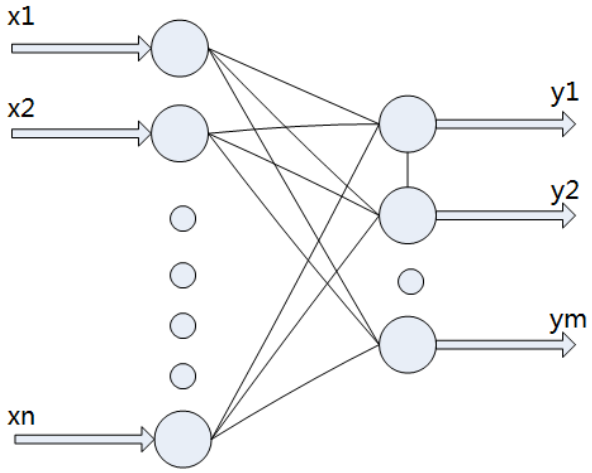


Figure 2.2 Two Layer Palm Network

The function or mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$ is memorized by the connection strengths between the neurons, measured by their synaptic weights. These weights are either 0 or 1, contained in a memory matrix $\boldsymbol{M}$. The learning process for the Palm network is essentially the generation of the memory matrix from the input and output patterns by a specific rule:

$$\mathbf{M} = \bigcup_{k=1}^{q} \boldsymbol{y^k} \times (\boldsymbol{x^k})^T \tag{2-4}$$

That is, if one multiplies the output pattern $\boldsymbol{y}^k$ (column vector) and input pattern $\boldsymbol{x}^k$ (row vector), you create an n by m weight matrix for pattern k. The complete memory matrix is the union of all these patterns' weight matrices. This union can be done with an OR operation for every element.

In the retrieval process, $\boldsymbol{x}'$, a distorted or noisy version of $\boldsymbol{x}^k$, is presented to the network. The expected output should be the corresponding $\boldsymbol{y}^k$. For one-step retrieval, all the neurons process their inputs and update the outputs simultaneously. The update rule can be written as:

$$\boldsymbol{y} = f(\mathbf{M} \cdot \boldsymbol{x}' - \theta) \tag{2-5}$$

Where f is the Heaviside function and θ is a fixed threshold for the neuron for a k-winner-takes-all strategy [28]. That is, only if the sum of all the neuron's inputs reaches the threshold can it output a 1. Otherwise, its output will be 0.

In the Palm model, both input vectors and output vectors are generally represented with a sparse coding technique to avoid the overlapping of patterns' one-entries, because conflicts in the memory matrix will result in the retrieval of spurious patterns. Though this technique may cost a large number of neurons, it is easy to implement in software or a digital system because a binary sparse matrix can be stored in a way that conserves memory.

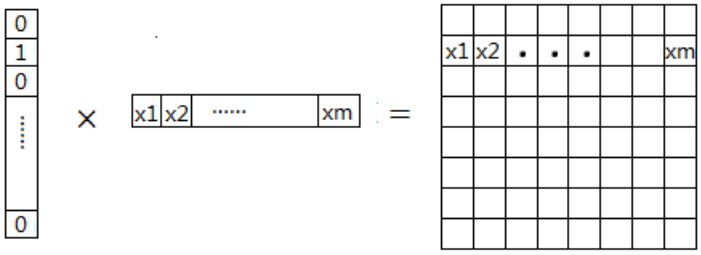


Figure 2.3 Unary Coding of the Memory Matrix for a Palm Network

One simple way to achieve sparse coding is that we can use a unary code for $\boldsymbol{y}$ (only one neuron outputs a 1). If the i^{th} element of $\boldsymbol{y^k}$ is a 1, then a copy of the input pattern $\boldsymbol{x^k}$ will be stored in the i^{th} row of the matrix, as shown in Figure 2.3. Thus each pattern takes one row, and does not overlap with others.

2.3 HOPFIELD NETWORK

The Hopfield network is a multi-loop feedback recurrent network that consists of a set of interconnected neurons which update their activation values. The neurons in this network are connected to all other neurons except to themselves, which means there is no self-feedback in the network. A simple three node Hopfield Network is shown in Figure 2.4. (x_1, x_2, x_3) and (y_1, y_2, y_3) are respectively input and output vectors, while w_{ij} represent the weight of connection between every two nodes. With delay on the feedback loops, each node updates its output depending on the input from other neurons, the weights, and its own activation function. The activation values are binary, usually (-1,+1) or (0,1). Given the recurrent feedback structure, the input vector and output vector are actually the same, and both consist of the state of each node, noted as s_i. The Hopfield Network uses the McCulloch–Pitts model to describe the behavior of a neuron [27]. Thus we have the update function of a node:

$$s_i = \begin{cases} 1, if \ \sum_j w_{ij} s_j > \theta_i \\ -1, \ otherwise \end{cases} \tag{2-6}$$

Where s_i is the state of node i, w_{ij} is the strength of connection weight between node i and node j, and θ_i is the threshold for the activation function. Updates in the Hopfield network can be performed in either of two modes:

(1) Synchronous: Update all the nodes simultaneously based on their latest states.

(2) Asynchronous: Only one node updates at a time. The update order is random or programmed.

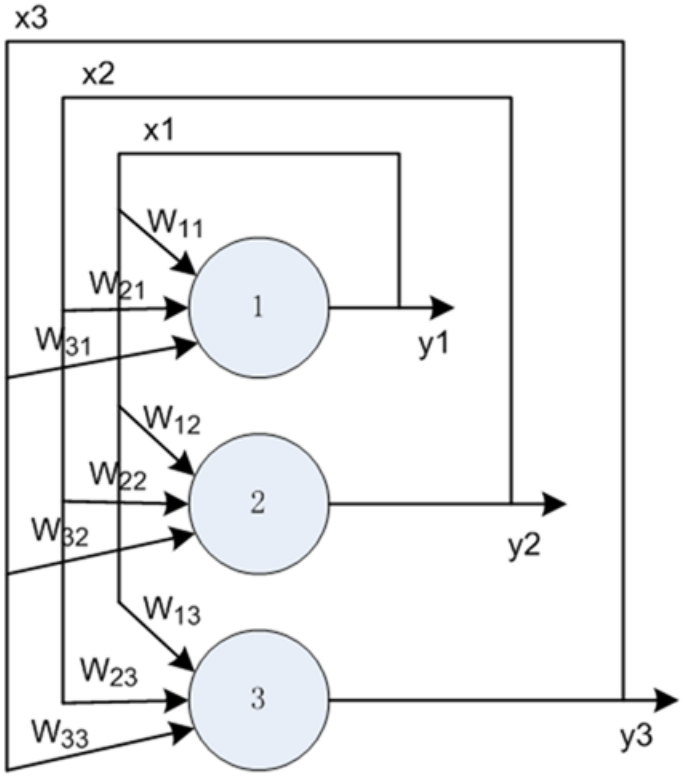


Figure 2.4 Hopfield Network with Three Nodes

The Hopfield network can function as an AM model. In the recognition process, a network is initialized by setting the values of all nodes as the input pattern. From this initial state, the states of nodes are repeatedly updated until the network converges to a stable state. In this final state, Update operations will not change on the output of the network.

Each state of the Hopfield network is associated with a scalar value as the "energy", defined as

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i \qquad (2\text{-}7)$$

Like potential energy in a simple dynamic system, any automatic updating in the Hopfield network can only make *E* lower or keep it unchanged unless we force the node state as a new initial state. Thus it is certain that network will eventually converge to a state which is a local minimum in the energy function under repeated updating.

Hopfield network memorizes patterns depending on the values of connection weights. Training rules for a Hopfield network determine how the weight matrix is generated. Many training rules are reviewed in reference [29]. In our discussion we use a Hebbian rule.

We build a Hopfield network as an AM model for simulation. In this case, the network is used to memorize two simple binary images (shown in Figure 2.5). Each image is an 8x8 binary matrix. Thus the network has 64 nodes.

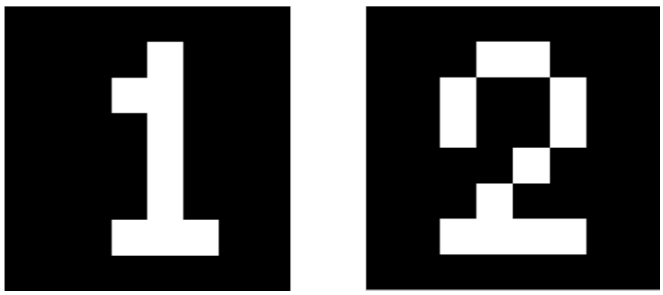


Figure 2.5 Two Binary Images for Testing a Hopfield Network

For training, we use the covariant Hebbian rule. Elements of the weight matrix can be obtained by the formula:

$$w_{ij} = \sum_{t=1}^{2} (2\xi_i^t - 1)\left(2\xi_j^t - 1\right) \tag{2-8}$$

where ξ_i^t is the i^{th} bit of t^{th} pattern.

As a simple test, we input a distorted version of the number "2" (the second pattern) and use the asynchronous update strategy to simulate this network. Figure 2.6 shows the whole retrieval process, a time series of snapshots of the output patterns. From this results we can see that the network retrieves a stored pattern closest to our input pattern.

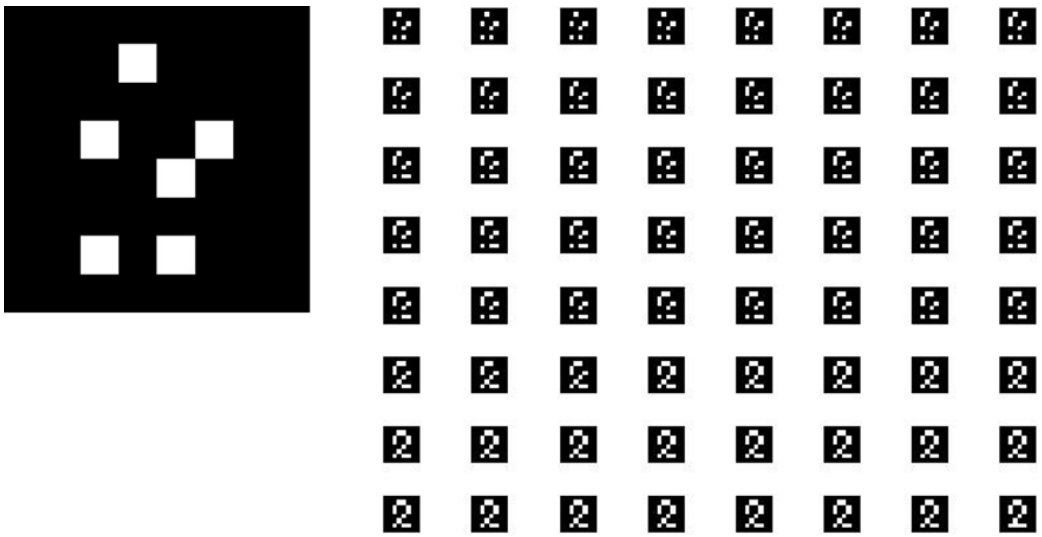


Figure 2.6 Snap Shots of Time Evolution in Pattern Retrieval Process

The Hopfield Network is a very effective AM model, and it explains the mechanism of AM from the perspective of a physical system, which is a very important contribution to implement AM function in real physical devices. The defect of this model is that many spurious patterns in the network usually lead to an unexpected result. In addition, the Hopfield network has a low capacity compared to the Palm network.

2.4 OSCILLATORY NEURAL NETWORK

Hoppensteadt and Izhikevich study weakly connected networks of neural oscillators near multiple Andronov-Hopf bifurcation points [30]. They propose a canonical model for oscillatory dynamic systems which satisfy four criteria:

(1) Each neural oscillator comprises two populations of neurons: excitatory neurons and inhibitory neurons;

(2) The activity of each population of neurons is described by a scalar (one-dimensional) variable;

(3) Each neural oscillator is near a non-degenerate supercritical Andronov-Hopf bifurcation point;

(4) The synaptic connections between the neural oscillators are weak.

Given these conditions, this canonical model can be represented by the function:

$$\frac{dz_i}{dt} = (\rho_i + i\omega_i)z_i + d_i z_i|z_i|^2 + \varepsilon \sum_{j=1}^{n} C_{ij}\, z_j \tag{2-9}$$

Where:

- $z = x + iy$ Complex oscillator variable
- ρ_i Damping of oscillators
- ω_i Natural frequencies of oscillators
- d_i Nonlinear factor, ensures stable amplitude
- ε Coupling parameter, typically small (weakly coupled)
- C_{ij} Coupling matrix, represent coupling strength between two oscillators similar to the weight matrix in previous models

This dynamic model was proved to be able to form attractor basins at the minima of a Lyapunov energy function by adjusting the coupling matrix though a Hopfield rule [27]. In other words, a network that consists of oscillators described by this canonical model can learn patterns as represented by phase errors and perform the functions of an associative memory.

We can illustrate Hoppensteadt and Izhikevich's theory by simulation. An example network model with 60 fully connected neurons is built for simulating the retrieval process. In this model, memorized patterns are the binary image of the numbers 0,1,2, that we showed in Figure 2.2. The structure of network is depicted in Figure 2.7.

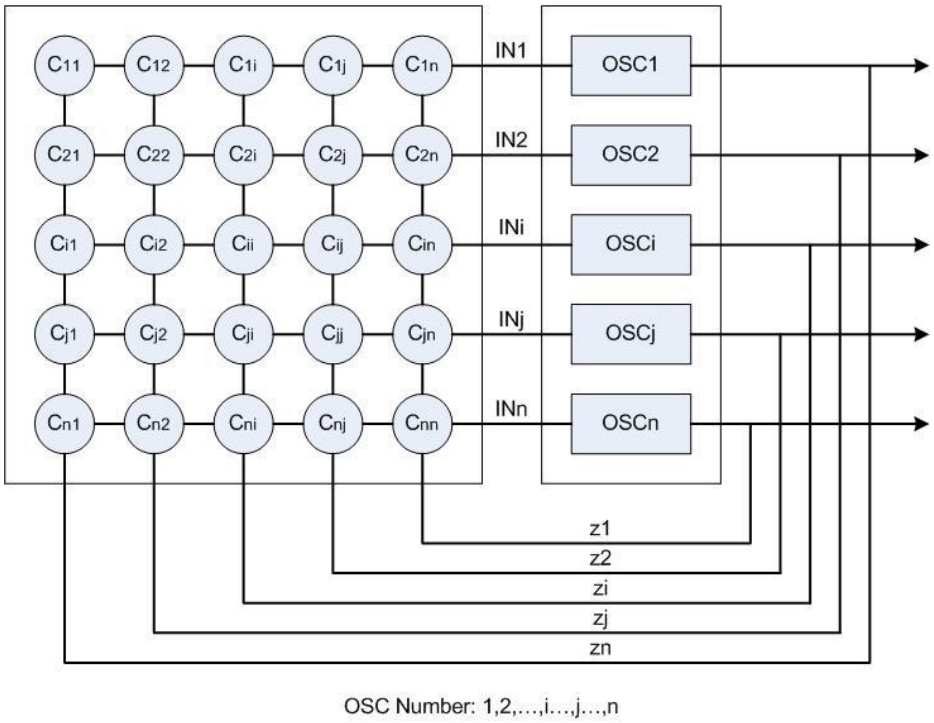


Figure 2.7 Oscillatory Neural Network Structure

The oscillator cluster (OSC1, OSC2, OSC3, …, OSCn) are coupled by a matrix of multipliers and adders that contain the coupling coefficients, shown as the C_{ij} box. From the list term of the equations above, the outputs are fed back to the network for the recurrent evolution process until the network state is completely stabilized at an attractor.

To perform the simulation, first, we set the coupling coefficient to initialize the system and to set the system in a state that represents the input pattern. Second we change the set of coefficients to represent the "learned" values and initiate the retrieval process. The figures below help us observe these two processes more directly. Initially, the oscillators each have a random relative phase. Then in the first phase, we introduce the input pattern, in this case a noisy "1." By step 10 of the simulation, the oscillators are holding that pattern. Figure 2.8 (a) shows the evolving process of complex oscillator variables and (b) shows the output phases for the 60 oscillators. Figure 2.9 interprets these phases as grayscale image at each step. In the second

phase, we change the C_{ij} matrix to hold the values for the learned patterns '0', '1', and '2'. Now the oscillators change their phase to accommodate the input pattern with respect to the stored patterns. The result is a stable pattern of phases that is close to the phase pattern for the ideal stored "1". The same plots as the first stage are shown in Figure 2.10 and Figure 2.11. We can notice that in both stages, the oscillator network always converges to an attractor formed by C_{ij}. It demonstrates that the oscillatory neural network based on the canonical model has the same function as the Hopfield network.

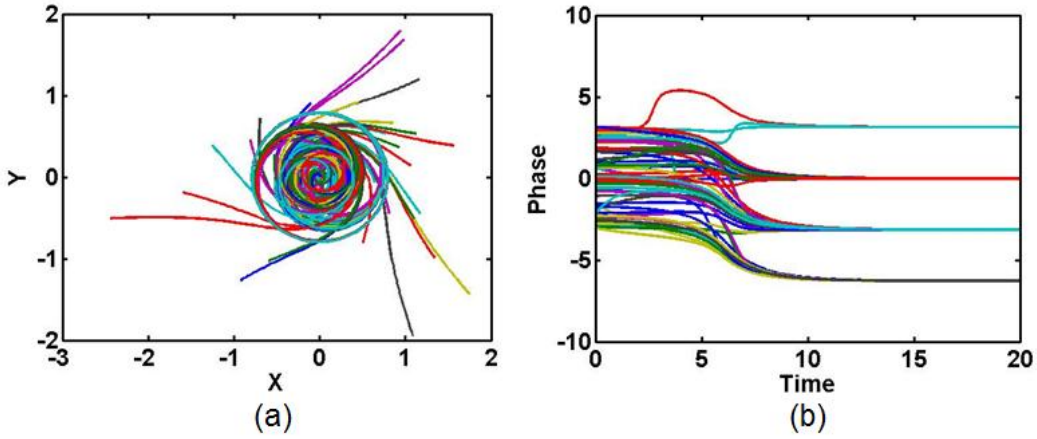


Figure 2.8 Evolution of Initial State

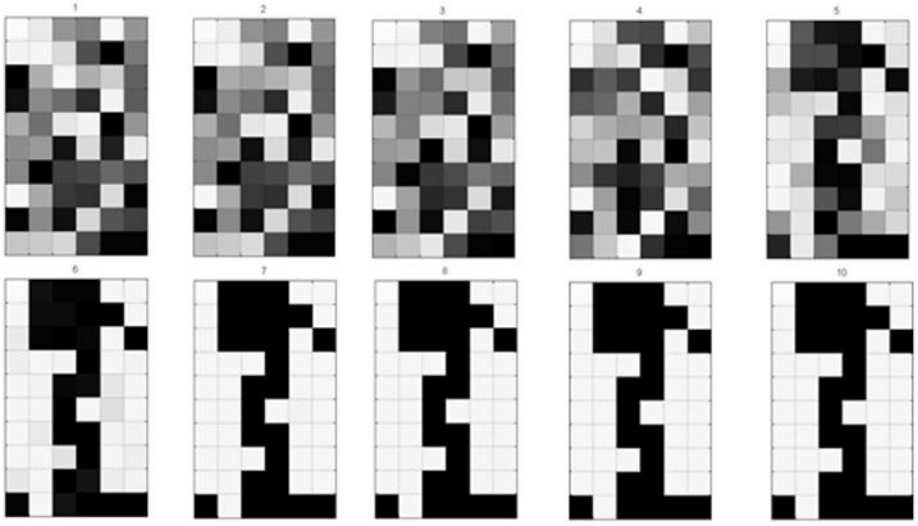


Figure 2.9 Phases as Grayscale Images in Evolution of Initial State

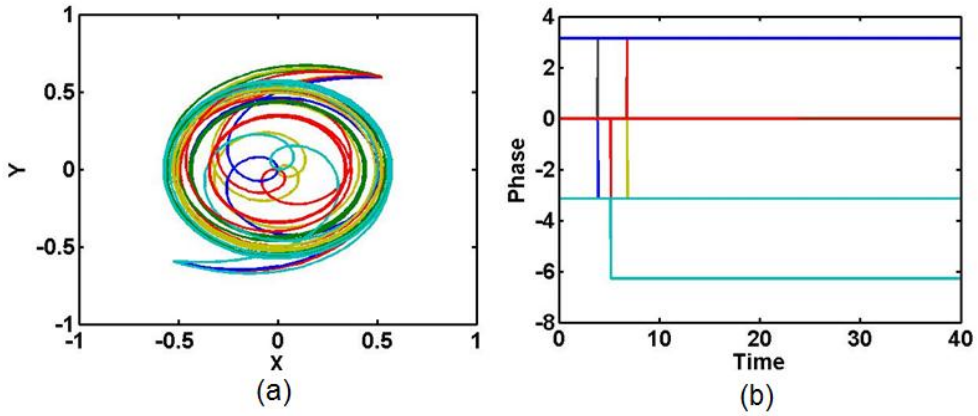


Figure 2.10 Evolution of Retrieval State

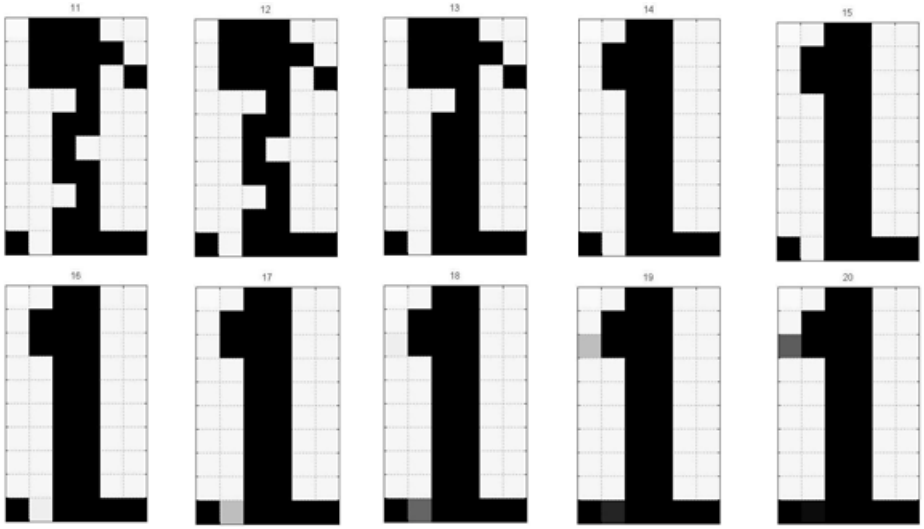


Figure 2.11 Phases as Grayscale Images in Evolution of Retrieval State

2.5 IMPLEMENTATION OF OSCILLATOR NETWORK

A phase-locked loop or phase lock loop (PLL) is a device that generates an output signal whose phase is related to the phase of an input "reference" signal. As Figure 2.12 shows, it consists of a voltage-controlled oscillator and a phase detector implemented as a multiplier and low pass filter.

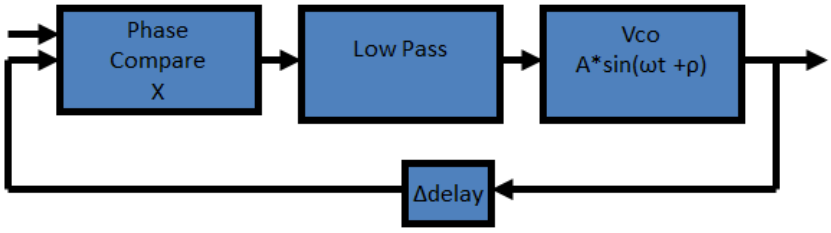


Figure 2.12 Structure of Phase Lock Loop

This circuit compares the phase of the input signal with the phase of the signal derived from its output oscillator and adjusts the frequency of its oscillator to keep the phases matched. The signal from the phase detector is used to control the oscillator in a feedback loop. Frequency is the derivative of phase. Keeping the input and output phase in lock step implies keeping the input and output frequencies in lock step. Consequently, a phase-locked loop can track an input frequency, or it can generate a frequency that is a multiple of the input frequency.

A network of PLL circuits can form a structure similar to the canonical oscillator network described above. The input signal of each PLL is the weighted sum of the outputs of every PLL, including itself. The only additional operation is that there is a $\pi/4$ delay in the feedback loop to make the input signals and reference signals inside the PLL orthogonal.

By simulation in Matlab, we can observe the properties of PLL network more directly. We have built the model with the following parameters:

- N=64, Oscillator Number
- f=1000, Natural Frequency of VCO (Hz)
- d=1/(f*100), Simulation Step Size
- T=0.5s Simulation Time
- sens=100, Sensitivity of VCO
- amp=1, Amplitude of VCO Output

- Kd=1, Gain of Phase Detector

Transfer Function of Loop Filter: $T(s) = \frac{10}{s+10}$

Step Iteration Rule: $\Delta V_c = f_c(\mathrm{PDout} - V_c(\mathrm{n}-1)),\ V_c(\mathrm{n}) = V_c(\mathrm{n}-1) + \Delta V_c$

As an example of the operation of this model we can store two patterns, the 8x8 image of '1' and '5', and observe the convergence behaviors of the array. In Figure 2.13 we use a random binary image (as a "very noisy" input), start the simulation and observe the evolution process of the phase behavior of the oscillators. The system converges to the closest stored pattern, which is '5' in this case.

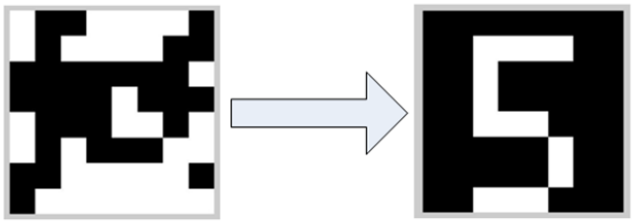


Figure 2.13 Binary Image Pattern Retrieval in a PLL Network

We plot the phase error between the oscillators in terms of their cosine value. In Figure 2.14 we see the time evolution of the phase error.

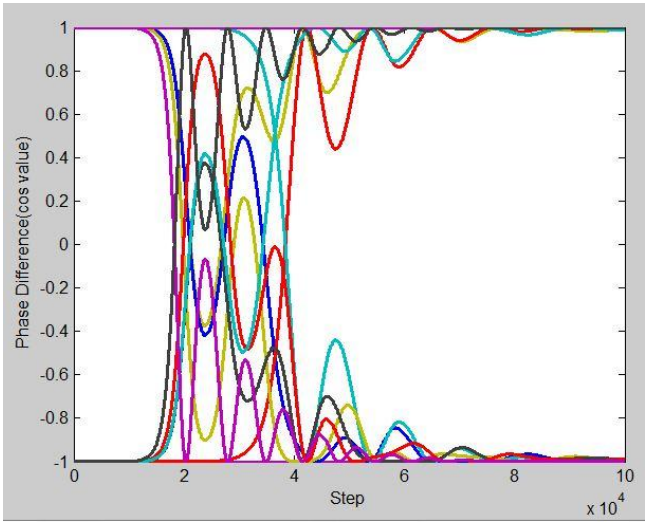


Figure 2.14 Evolution of the Phase Error in a Binary Image Pattern Retrieval Process

In a second test, we give the system a random gray scale image, which means that the initial condition is not fixed to binary values. Similarly, Figure 2.15 and Figure 2.16 shows the retrieval process.

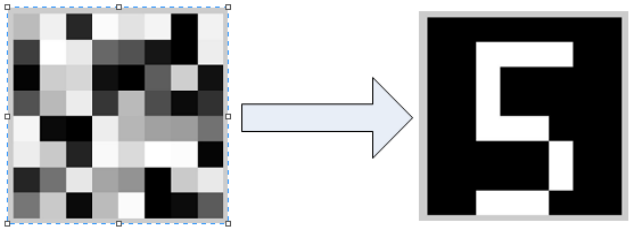


Figure 2.15 Grayscale Image Pattern Retrieval of PLL Network

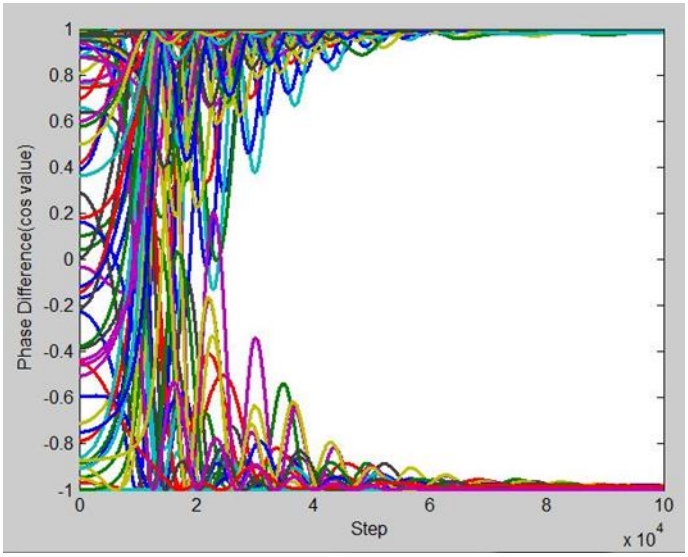


Figure 2.16 Evolution of the Phase Error in Grayscale Image Pattern Retrieval Process

From the experiments above, we suggest that the Phase-Locked Loop, as both a model and an implementation for the basic neuron of oscillatory neural network, is able to behave as a canonical oscillator model, with some merits like fewer spurious states and ease of fabrication in CMOS. However, it is costly in terms of hardware resources compared with several novel devices presented later. It also has a slow converging speed. In fact using the Palm coding rule, the PLL network cannot guarantee the convergence.

Compared to the PLL, oscillator based on neuron MOS [31] is a much better choice for building an AM model. Neuron MOS transistor has multiple floating gates. In a simple CMOS ring oscillator, an inverter consisting of neuron PMOS and NMOS transistors have an input voltage capable of controlling the delay of signal propagation and thus change the oscillating frequency. Based on these oscillators, we can build an AM circuit for each stored pattern. This circuit reads the difference between the input pattern and stored pattern, and then outputs the degrees of match as the result of a search for the "nearest" winner pattern.

The spin torque nano-oscillator (STNO) is an emerging nano-device that can also be used to implement oscillator AM model. STNOs work on injection of high density current into multiple magnetic thin layers. Current-driven transfer of angular momentum between the magnetic layers in such a structure results in spontaneous precession of the magnetization once the current density exceeds the stability threshold. This magnetization precession generates microwave electrical signals as a result of an effect called giant magneto-resistance (GMR) that occurs in such magnetic multilayer structures [32]. This type of oscillator is tunable through applied dc current or magnetic field and exhibit high Q-factors in excess of 10,000 at 25 GHz. In addition, STNO demonstrates the functionality associated with VCOs using STNOs, including frequency modulation and injection locking [33][34]. Given these characteristics, STNOs can couple each other by mutually locking on phase or shifting each other's frequency. Oscillator Neural Network model based on STNO needs further research in circuits and devices, which beyond the scope of this thesis.

Different from the PLL based Oscillator Neural Network, a network model composed of nano-oscillators needs no complex feedback connection matrix because these oscillators are coupled to each other in a cluster, instead of two, as shown in Figure 2.17. Each oscillator can

influence all the other ones simultaneously without coefficients computed from multipliers and adders.

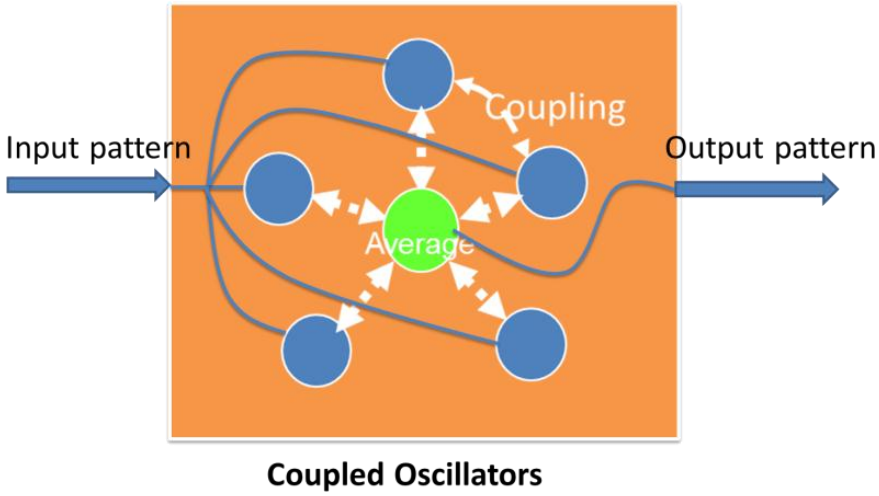


Figure 2.17 Coupled Oscillators Cluster

Although it is very difficult to couple multiple nano-oscillators together and we do not yet know about the capacity of an AM model in this structure, it can be designed as a differentiator that output the degree of match from the input of any two vectors. The degree of match is determined by the energy function of the oscillator cluster as we mentioned. An AM model can be built by assembling many clusters like this. And we can get the "nearest" pattern by comparing their outputs.

2.6 SUMMARY

In this chapter, we briefly introduce some of the previous work of associative memory. Among a number of models, we analyze three relevant ones in detail. They are the Palm Network, the Hopfield Network, and the Oscillatory Neural Network. Their advantages, disadvantages and relationships to each other are discussed. We also build and simulate Matlab models of the

Hopfield Network and the Oscillatory Neural Network. These two models have similar behavior of dynamics. They both form attractor basins at the minima of their energy function and make the initial state converge to the nearest one. Then we use the model of a real device, PLL, to implement Oscillatory Neural Network and successfully verify its functionality as an AM system. PLL is slow and large. Therefore, we introduce two emerging device techniques, neuron MOS and Spin torque nano-oscillator as possible alternative devices for AM design in the future. They are fast and scalable, and also consume little area.

3.0 HIERARCHICAL ASSOCIATIVE MEMORY MODELS

In this Chapter we introduce our exploration on hierarchical architectures for assembling many AM units. We begin from building a simple model and improve our design step by step with modeling and simulation.

3.1 MOTIVATION OF HIERARCHICAL MODEL

We have shown that weakly coupled oscillatory neural networks are capable of memorizing multiple patterns and retrieving the "closest" one to an input pattern. Taking this idea one step further, we believe that oscillatory neural networks can also perform the functions of the Palm model and form a complete associative memory system.

Nevertheless, there still exist several technical problems when novel oscillatory devices are used to form large scale neural networks. Full connection within the associative memory requires the full and tuned connection of oscillators within the network, which is difficult for coupling between nano-scale oscillators. Even if we can build a large "flat" neural network, the maximum efficiency of a fully-connected Palm model can only reach 35% [28]. Compared with a biological neural system, this type of model has a very low efficiency.

On the other hand, hierarchical structures have been identified in the neocortex of human beings and this area of the brain is believed to perform the functions of memory retrieval and

logical inference [35]. We can hypothesize that a large number of associative memories are organized hierarchically in this structure. Therefore, we have investigated the design of a hierarchical associative memory system made by connecting Palm models.

3.2 A SIMPLE HIERARCHICAL AM MODEL

Figure 3.1 illustrates a three-level hierarchical model that can memorize 8x8 binary images (64 bits in total). As the figure 6.1 shows, each module (X1…X4, Y1,Y2 and Z1) represents one Palm network with 16 input neurons and 8 output neurons. Every module in a higher level receives the output of two blocks from the lower level. The training patterns and the test patterns are sliced into 4 pieces (shown with red lines) and are respectively fed into the 4 "X" modules in the bottom level.

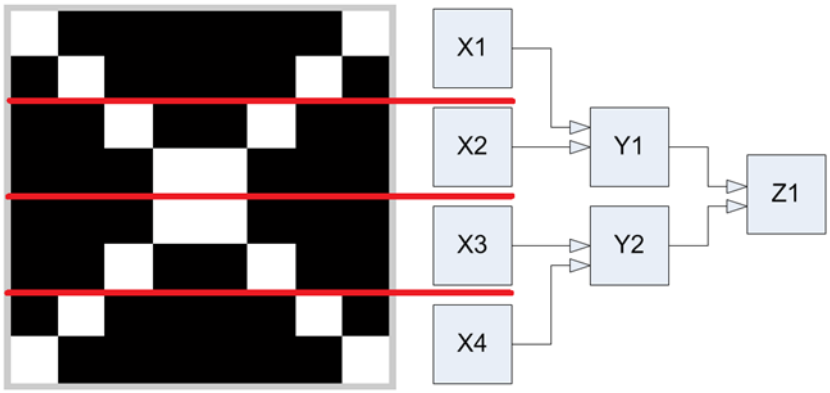


Figure 3.1 Three-level hierarchical structure and Receptive Field

The hierarchical model is trained from lowest level (X level) to the highest level (Z level) using the Palm model's training rule, the union of matrix multiplications. After each lower level is trained, it feeds its output vectors as the training vectors to the next higher level. Therefore, eventually, every Palm module learns the patterns of its receptive field (the fraction of the pattern

under it in the hierarchy) and builds the input-output mapping function to generate a corresponding output vector. The pattern retrieval task, like the training task, is processed sequentially. In this case, we put a distorted image pattern into the X level. The Palm modules at the X level will output unary codes to the Y level, which will then generate output codes, and the module at the Z level will recognize the whole pattern based on the Y level outputs. To more closely model the behavior of a hypothetical Palm network built from oscillatory neural networks, we adjusted the Palm model in its retrieval process. Instead of using a Heaviside function and threshold to update the output neurons, we take a "winner-take-all" (WTA) strategy. In this technique, $\mathbf{M} \cdot \boldsymbol{x}'$ is a vector we can call the degree of match (DoM), which indicates the similarity between the input pattern and each memorized pattern. And, the pattern with the highest DoM is the winner. All other patterns will be "losers." A winner is randomly picked when multiple patterns have the same highest DoM.

In our tests, we used a unary code for the output pattern of each Palm network and train the system with 4 simple image patterns label P1 to P4 in Figure 3.2.

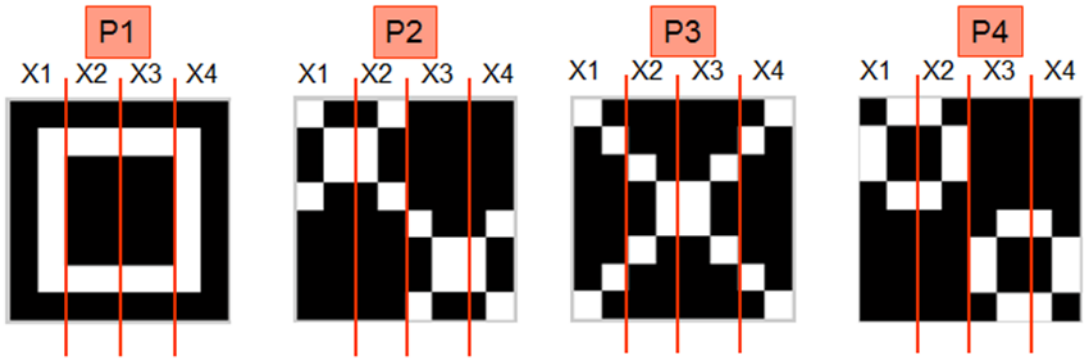


Figure 3.2 Binary Image Patterns for First Level of Hierarchical Palm Network

To test that the hierarchical model would function as expected, we test many random binary images and observed the retrieval process of each Palm module. One illustrative example of this retrieval process is shown in Figure 3.3 and Figure 3.4.

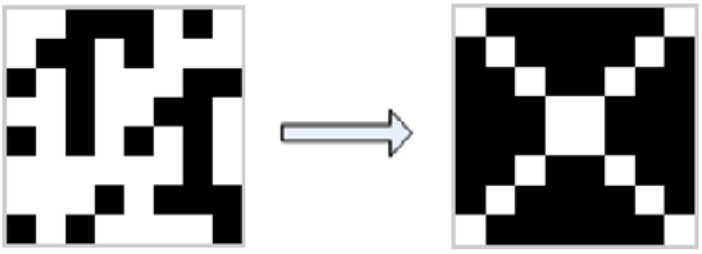
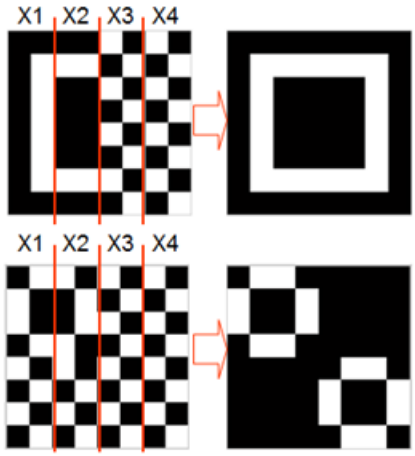


Figure 3.3 Input Pattern and Output Pattern in Function Test for Hierarchical Model

AM	P1(dom)	P2(dom)	P3(dom)	P4(dom)	winner
X1	5	3	2	3	p1
X2	2	1	3	2	p3
X3	2	2	3	4	p4
X4	1	1	2	2	p3/p4
Y1	1	0	1	0	p1/p3
Y2	0	0	1	1	P3/p4
Z1	0	0	2	0	p3

Figure 3.4 Function Test Retrieval Process

After the hierarchical model's ability to retrieve patterns was tested, we still wante to test if the patterns retrieved met our expectations of finding the "right" pattern, which is in agreement with our human perception. Hence, we developed a "robust test" where we used an input pattern with some part of the pattern exactly the same as a trained pattern and some part of the receptive field distorted. Figure 3.5 presents the recovery of two such distorted patterns and the DoM of each Palm network as the recognition process proceeds.



	Test1:DOM				Test2:DOM			
AM	p1	p2	p3	p4	p1	p2	p3	p4
X1	6	2	2	1	2	0	0	4
X2	4	1	0	4	2	0	0	4
X3	2	2	2	2	2	2	2	2
X4	3	2	2	2	3	2	2	2
Y1	2	0	0	0	0	0	0	2
Y2	1	0	0	1	2	0	0	0
Z1	2	0	0	0	1	0	0	1
Pattern	p1				(random) P4			

Figure 3.5 Robust Test For Hierarchical Model

These simulation results show that this Hierarchical AM model can memorize and retrieve patterns with similar results to a large single associative memory.

We can make a comparison between this hierarchical model and traditional single AM. Assume they have the same size of receptive field, an n by n square. If we use the same design for hierarchical model, we can estimate the features below (Table 3.1).

Table 3.1 Comparison between Hierarchical AM Model and One Single AM

	Hierarchical AM Model	One Single AM
Model	Palm, Unary Code	Palm, Unary Code
Receptive Field	n^2	n^2
Number of Neurons	$\frac{7}{4}n^2$	n^2
Pattern Capacity	$\frac{1}{4}n^2$	n^2
Connections	$\frac{7}{16}n^4$	n^4

Although hierarchical model takes more neurons in that it has multiple networks, it uses much less connections, which is a very important advantage for an oscillator neural network. However, we still cannot guarantee the hierarchical model has the same retrieval performance as

a single AM. Nonetheless, the connections of a single AM can be used to build up a hierarchical model with four same size layers. With this scale we may find a method to improve the retrieval performance.

3.3 SYSTEM IMPROVEMENTS

3.3.1 Pattern Conflict

During the pattern retrieval process, when the lower level's AM units output different patterns with the same DoM, the system sometimes performs poor recognition due to the random choice for tie breaking. In the case shown in Figure 3.6, the correct pattern should be P1. But the AM units in the second level and top level randomly pick a pattern between P1 and P3 because the same DoM in the local AM unit. The AM units in the same layer are independent and cannot communicate with each other. This phenomenon reflects the weakness of the hierarchical AM model structure. Without more connections between networks, associative information of different receptive fields is lost from the lower levels to the high levels.

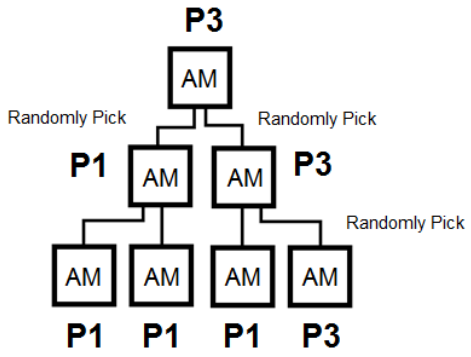


Figure 3.6 Example of the Pattern Conflict Case

3.3.2 Difficulty in Information Abstraction

In an ideal hierarchical structure, the higher level input vectors are the concatenation of lower level modules' input vectors to prevent information loss. Thus the size of network increases as one goes to higher levels. Unfortunately, the oscillator AM module cannot be very large due to the practical consideration of implementing large networks oscillators. Therefore if we keep all the AM units the same size, the output patterns from the lower levels to the higher levels have to be abstracted into shorter vectors. Therefore the original information of the input pattern is lost.

3.3.3 Capacity Problem

In real applications, a hierarchical AM system may be required to memorize a large number of patterns. Since we cannot store all patterns in a single AM module, and we have fewer AM units in the higher level of a hierarchical model, the capacity problem becomes increasingly serious. Given the fact that the top level has only one AM unit, it cannot store as many patterns as the bottom level can. In addition, for an oscillator network, more patterns means lower speed and higher error rate.

3.3.4 An Improved Hierarchical Structure

We still keep the hierarchical design but change the receptive field of each block, as Figure 3.7 shows. Consider the input image as a 64x64 matrix, every module receive the input of a 2x2 square area of lower level, instead of 1x2 stripe. The bottom Level (L1) slices the input image

into 16x16 blocks and each one has 4x4 pixels. This design can resist distortions and noise in the image pattern much better.

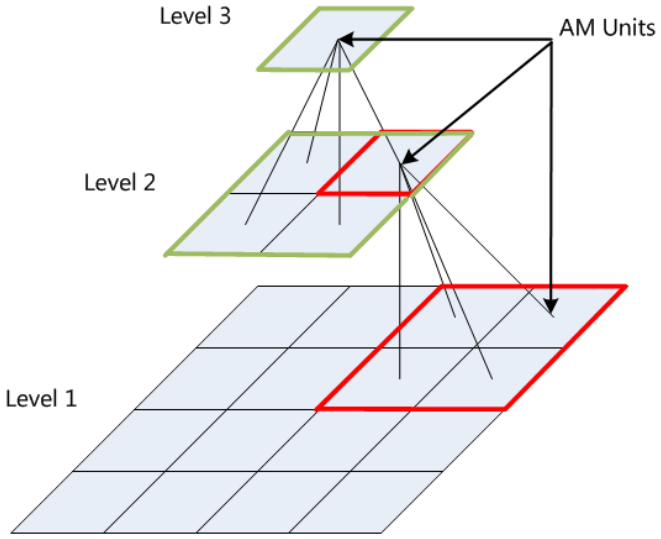


Figure 3.7 Pattern Representation between Layers

To keep every AM module the same size, the output vectors of 4 AM modules are translated into a binary code and concatenated to the input vector that is fed to higher levels. In this case, each AM module can memorize 16 16-bit binary vectors. Their output are coded into 4-bit vector and fed to the next level, shown in Figure 3.8.

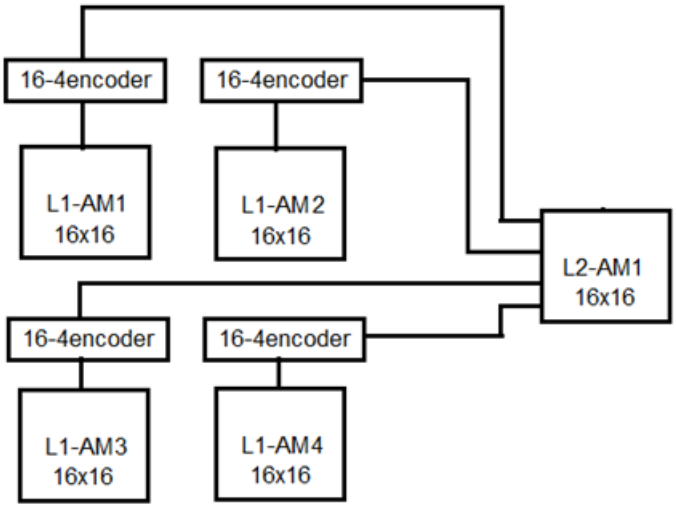


Figure 3.8 Encoders between Two Layers in the Hierarchical Model

The function of 16-4 encoder is to convert the output of the 16-bit unary code into a 4-bit binary code. The original 16-bit unary code indicates the index of the output pattern. To obtain the DoM, we use the XNOR operation instead of the multiplication in the Palm model. Figure 3.9 provides an example how the AM module recognize pattern.

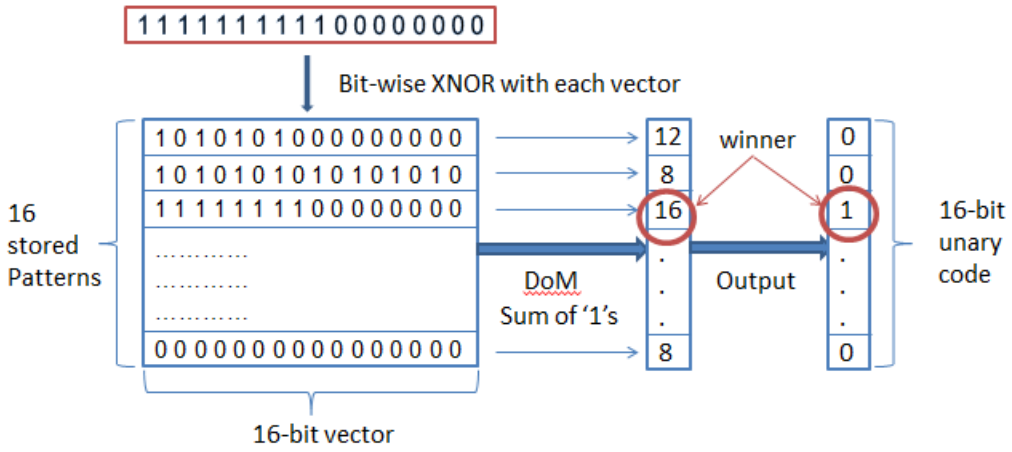


Figure 3.9 AM Units Function in the Hierarchical Model

3.4 SIMULATION AND EXPERIMENTS

To test the performance of our system, we randomly pick 16 64x64 images that were converted from JPEG color pictures, which are presented in Figure 3.10. Two sorts of input patterns are used for the test, the first one is the original image with added noise, and the second one is a combination of different parts of two images.

Figure 3.10 Binary Image Set in Pattern Retrieval Test

3.4.1 Noisy Pattern Retrieval Test

In the noisy pattern retrieval rest, we randomly choose a percentage of image pixels and flip their value (from 0 to 1, or from 1 to 0). This sort of noise is added to every image in the data set. Figure 3.11 shows us 25% noise on an image, which is hard to recognize for human eyes.

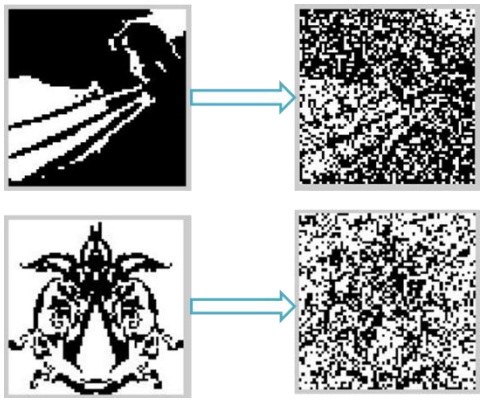


Figure 3.11 25% Noise in Noise Retrieval Test

We vary the noise percentage from 0 to 50%, and run a simulation 500 times for each pattern under each noisy input. We observe the hit rate of each simulation. Figure 3.12 gives the curves of hit rate for 16 patterns.

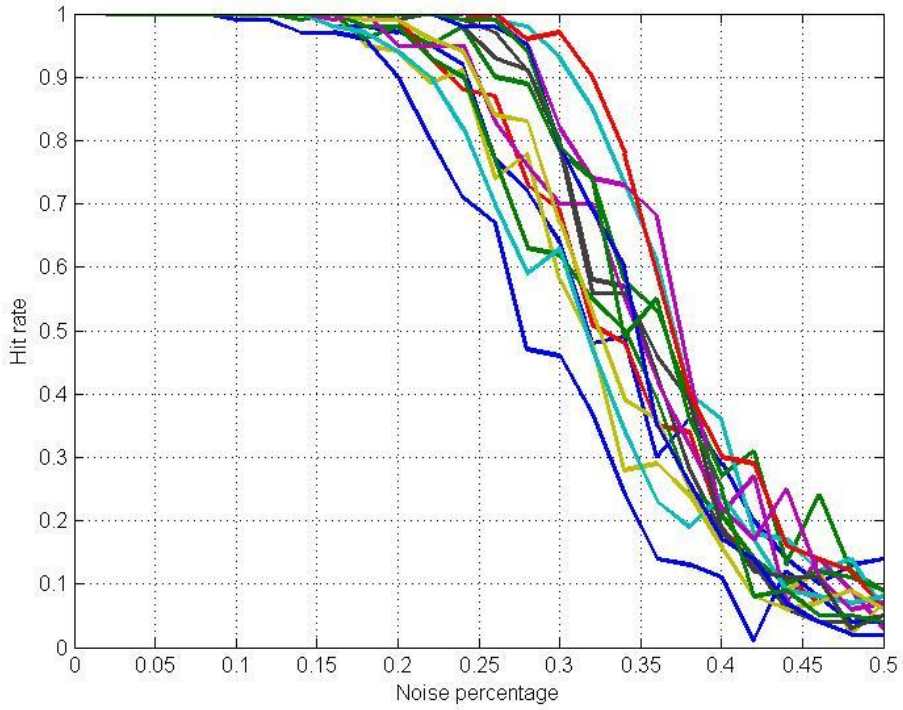


Figure 3.12 Hit Rate vs. Noise degree in Noise Retrieval Test

3.4.2 Pattern Conflict Test

To test if our new design solves the pattern conflict problem, we intentionally combine 2 images by replacing some columns of one image by another's. In the simulation, we vary the replacement ratio from 10% to 50%, and use 16 different patterns to interfere with the 4th pattern. The curves of hit rates and replacement ratio are shown in Figure 3.13. From this result we find that this model reduces pattern conflicts.

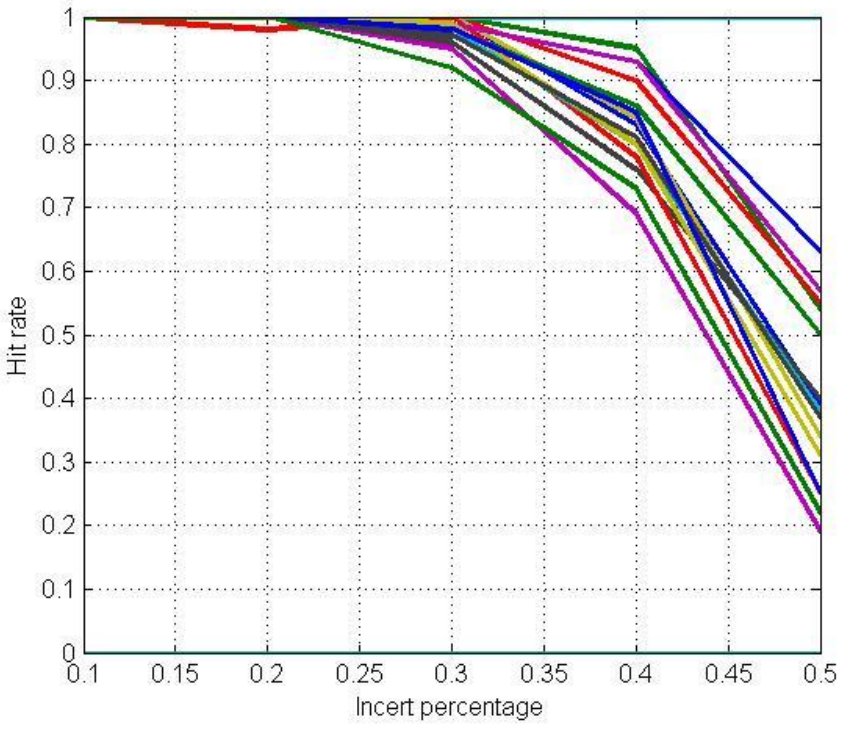


Figure 3.13 Replacement Ratio vs. Hit Rate in Patten Conflict Test

3.4.3 Pattern Storage

Figure 3.14 (a), (b), (c), (d) present us the use of storage space from level 5 to level 2. We did not give level 1 because we know the top level has only one AM unit and store 16 patterns. The heights of the 3-D bars indicate how many patterns an AM module memorized for this data set. The maximum is 16, determined by the AM unit. The x and y axis show the position of a module in the AM array of each layer. As we expected, AM units in the lower levels store fewer vectors than AM units in a higher level because some images share the same pattern in a small receptive field. Since level 2 and level are both running out of storage, we cannot memorize more patterns without changing the size of AM unit. Hence, we failed to solve the capacity problem.

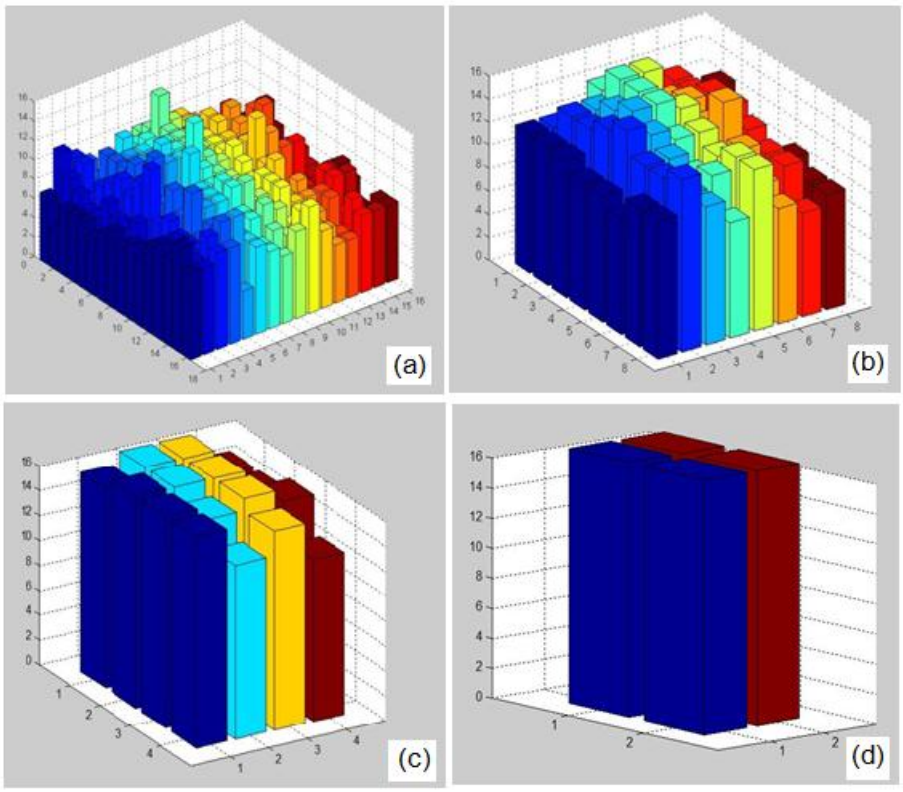


Figure 3.14 The Memory Usage of each Layer of the Hierarchical Model

3.5 DYNAMICAL LEARNING

In many applications for associative memories, the process of learning all the patterns followed by recognizing inputs may not always happen sequentially. For human beings, these processes actually occur simultaneously [36]. By memorizing and recalling many times, we can usually master skills and knowledge better by the act of reinforcement. It is also important for AM systems to change the set of learned patterns over time. If we fix the knowledge of an intelligent system for its whole life span, it would not work well in a changing of environment.

In the particular case of associative memories, we also face the problem of limited capacity for patterns. Thus, it is beneficial to adjust the area of the attractor basins dynamically,

based on the repeated process of training and retrieval. The more times a pattern is used for training or is successfully retrieved the higher weight (as a likelihood) it should have in future retrieval tasks.

To achieve this function, we add a row of counters to keep a priori probability weight for each stored pattern in every AM unit, and increase or decrease pattern's a priori probability by adjusting the weight vector after each operation. For example, when the occurrence of pattern1 increases, we assume its probability of occurring in the future will be increased and thus we increase its weight. During recognition, when two patterns have the same DoM, the one with higher weight will be the passed to the output as the winner. At the same time, the weight of that pattern is increased.

To test this idea, we simulated the retrieval of an image which has the same DoM for pattern2 and pattern4 from Figure3.2 in a pattern retrieval test, as shown in Figure 3.15. Here we first simulate training the network with repeated various ratios of P2 and P4 by simply changing the ratios of the weights associated with patterns 2 and 4. Then we follow with a retrieval test that uses a weighted random number to pick between the patterns.

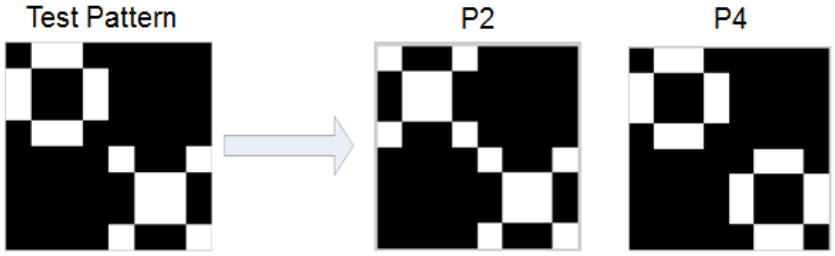


Figure 3.15 Dynamic Learning Test For Hierarchical Model

The simulation results of this test are shown in Figure 3.16. The x-axis is the ratio of training times of two pattern, while the y-axis is the occurrence of P2 in 500 retrieval tests. This

experiment shows that the statistical nature of the input (the frequency of occurrence) does influence the prior condition of retrieval process through dynamic learning.

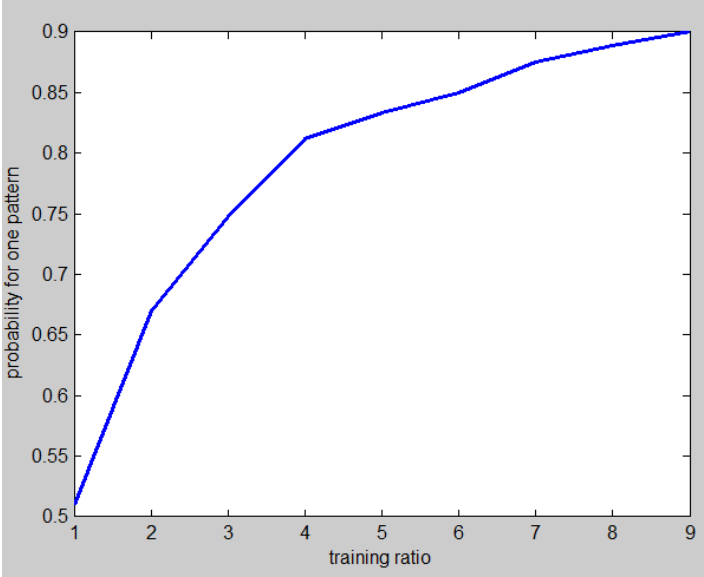


Figure 3.16 Dynamic Learning Test Result of Hierarchical Model

3.6 CONCLUSION

In this Chapter we give our motivation to build an AM system hierarchically, and then proposed a simple first prototype that consist of three levels of AM units. We use the Palm network with binary unary coded patterns to model the function of an AM unit based on oscillator neural network. This first model is tested and discussed. It functions as an AM system and reduces the connections of a single AM unit. We discovered its flaws and improved the structure and pattern representation. The improved hierarchical AM model for binary images has strong resistance to noise in patterns and is capable of retrieving heavily distorted patterns. It also reduces the pattern conflict in the retrieval process to a great extent, solves the information abstraction problem and keeps the AM module size unchanged. We also explore its potential for dynamical online

learning. However, it failed to solve the capacity problem. The capacity of AM unit is constrained exactly by the size of AM itself. Finally, we explore the potential of dynamical online learning. We add a special factor on the degree of match to represent the prior probabilities of patterns during retrieval and training. Simulation results indicate that the frequency of event occurrence does influence the retrieval process.

4.0 N-TREE MODEL

In this Chapter, we introduce the N-Tree model developed in collaboration with colleagues from Intel Corporation [37][38]. Also, a search algorithm called branch and bound search is employed to improve the performance of this model.

4.1 N-TREE MODEL DESCRIPTION

The Hierarchical AM model we proposed in the previous chapter does not solve the capacity problem because it only partitions the pattern vectors and compresses them. The input pattern still has to be compared with all the memorized patterns for the nearest neighbor search. Therefore, it is difficult to scale and cannot effectively handle large data sets in high dimension spaces. Since both the number and length of vectors grows. Given the fact that a tree structure always contains the least information in its root node and the most information in its leaf nodes, we reconsider the organization of data set and propose another model called the N-Tree model.

4.1.1 Model Structure

Assume there is a pattern set with m patterns, $\{p_1, p_2, \cdots, p_m\}$, required to be memorized by an AM system. When the number of patterns, m, is very large, it becomes difficult for a single AM

unit based on oscillators to hold all the patterns. As discussed previously, this is due to both fabrication and interconnects symmetry constraints. In order for the oscillators to work correctly as a pattern matching circuit, the oscillators need to be matched in performance and the interconnect needs to be symmetrical. Both of these goals can only be reliably achieved if we keep the clusters of oscillators relatively small. Therefore, as a solution to this problem, we use a hierarchical AM model that organizes multiple AM units into a tree structure.

If each AM unit can only store n patterns, while $m > n$, then the m patterns can be clustered hierarchically into an n-tree structure and stored in an AM model with the corresponding architecture. In this tree structure, every node is an AM unit and has at most n children nodes. Only leaf nodes (nodes that have no children) store specific patterns from the original input set: $\{p_1, p_2, \cdots, p_m\}$. The higher nodes are required to memorize the information associated with different levels of pattern clusters. Thus, every search operation, to retrieve one pattern, results in a path in the tree, from the root to a leaf. During the retrieval process, the key pattern is input recursively into the AM nodes at different level of tree and finally the pattern with the highest degree of match is output.

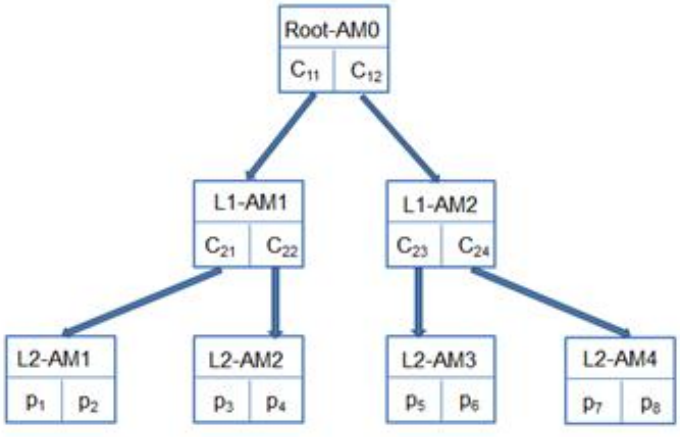


Figure 4.1 Three Layers N-Tree Model with Fanout=2

For example, if we take the simple case where, $m = 8, n = 2$, this case forms the binary tree AM model shown in Figure 4.1. At the root node, patterns are clustered into two groups, $\{p_1, p_2, p_3, p_4\}$ and $\{p_5, p_6, p_7, p_8\}$, respectively, associated with two representative patterns: C_{11} and C_{12}. There are several ways that the *C* patterns can be generated during training, as explained below. From the second level, these two groups of patterns are split again into four groups and are stored in four AM units in the third level. In effect, each AM node has an associated pattern, *C*, stored in its parent node which represents the patterns in its local group.

4.1.2 Training Process

Once the hierarchical structure of AM model is fixed, the next question is how to organize the patterns into a tree structure and which features can be used for the recognition at each node? These questions determine the method of training and pattern retrieval. Notice the fact that the AM unit always outputs the pattern with the highest degree of match, namely the one nearest to the input pattern in the data space. For this work, we use a hierarchical k-means clustering algorithm.

K-means clustering [39] is a method of cluster analysis which aims to partition n patterns into k clusters in which each pattern belongs to the cluster with the nearest mean. Since these are multi-dimensional means we also refer to them as “centroids”.

The steps of the algorithm can be summarized as:

1. Randomly select k data points (vectors) as initial means (centroids)
2. Assign every other data point in the training set to its closest mean (Euclidian distance) and thus form k clusters

3. Recalculate the current mean of each cluster based on all the data points that have been assigned
4. Repeat (2) (3) until there are no changes in the means

Hierarchical k-means clustering is a "top-down" divisive clustering strategy. All patterns start in one cluster, and are divided recursively into clusters by clustering subsets within the current cluster as one moves down the hierarchy. The clustering process ends when all the current subsets have less than k elements, and they are non-dividable. This algorithm generates a k-tree structure that properly matches our AM structure. The internal nodes are centroids of each cluster in different levels and the external nodes (leaf nodes) are exact patterns. During the clustering process, some clusters may become non-dividable earlier than others, and therefore have higher positions in the tree than the other leaf nodes.

Consider the previous example, the first clustering generates two clusters, $\{p_1, p_2, p_3, p_4\}$ and $\{p_5, p_6, p_7, p_8\}$ with centroids C_{11} and C_{12}. Then these two clusters are split to four sub-clusters, $\{p_1, p_2\}$, $\{p_3, p_4\}$, $\{p_5, p_6\}$ and $\{p_7, p_8\}$. Their centroids are C_{21}, C_{22}, C_{23} and C_{24}. Since the number of patterns in one cluster is less than the threshold, we set (n=2), the clustering process ends. After the centroids and patterns are written into each AM node correctly, the training process is finished.

4.1.3 Recognition Process

The pattern retrieval process starts from the root node. When we input the pattern that needs to be recognized, the AM unit at the root node will output the nearest centroid as the winner. This centroid will lead to a corresponding node in the next level and the system will repeat the recognition process until the current node is a leaf node. The final output is a stored pattern

rather than a centroid. Considering the same example as above, assume we have a test pattern p_t, which is closest to p_5. The retrieval process is:

1. Input $p_t$to the root node AM0, the output pattern is C_{12}, which points to the AM2 of level 1, with the cluster $\{p_5, p_6, p_7, p_8\}$
2. Input $p_t$to L1-AM2, the output pattern is C_{23}, which points to the AM3 of level 2, with the cluster $\{p_5, p_6\}$
3. Input $p_t$to L2-AM3, the final output pattern is p_5

In general, the N-Tree model reduces the steps of comparison by locating the input pattern in different clusters. Also, this architecture makes an AM system capable of scaling up or down as the data set changes. We only need to vary the tree structure of the data set.

4.2 TESTING AND PERFORMANCE

We implemented this N-Tree Hierarchical AM model for image recognition tasks. In this chapter, we use two image data sets to evaluate the performance of our model. The first data set is from the ATT Cambridge Image Database [40]. This data set contains 400 grayscale images of human faces. These images are photos of 40 people with 10 views for each person. The original images have 92x112 pixels with 256 gray levels.

We convert the image size into 32x32 and normalize the intensity of all the images by the following method:

1. Calculate the means of all pixels for each image, $(x_1, x_2, \cdots, x_{400})$
2. Calculate the mean of all images $(x_1, x_2, \cdots, x_{400})$, $\mathrm{M} = \frac{\sum_{i=1}^{400} x_i}{400}$

3. For i^{th} (1<=i<=400) image, every pixel is multiplied by the factor $\alpha = \frac{M}{x_i}$

For the tests in this Chapter, we do not use any image processing techniques, like feature extraction, to preprocess these images. Rather, the pixel values of an image, or centroid, are directly stored as one vector in our AM model. The purpose of the normalization of size and intensity is to make all pattern vectors fit our model.

In these tests we define the size of our AM unit as 16x1024. One AM unit can store 16 images and each image has 1024 (32x32) pixels. Thus the nodes of the tree structure can have at most 16 child nodes, instead of two in the example above. The definition of the algorithms in terms of both the training and recognition process remain unchanged.

In this task, the N-tree model is required to recognize the person in the input image after training. We use a hierarchical version of the ARENA algorithm of [41], which performs nearest neighbor search with a reduced-resolution face image. For evaluation, first, we split the data set into two parts, a training data set and a test data set. The partitioning is based on the subjects. For each subject, we pick 9 images for training and 1 for test. Thus there are 360 images in the training data set and 40 images in the test data set.

After the model has memorized all the images, we input images of the test data set sequentially. If the output image and the input image belong to the same subject (from different views) we say the recognition is successful, otherwise we call it a failure. The "hit rate" of this simulation is computed based on the 40 test results, one per subject.

In order to evaluate the system's performance, we use a cross-validation technique [42]. For each run, we randomly pick images for training and for test of each subject such that the model trained and tested on different data set every time. One example of this partitioning is shown in Figure 4.2. We performed 1000 runs each with a different partition.

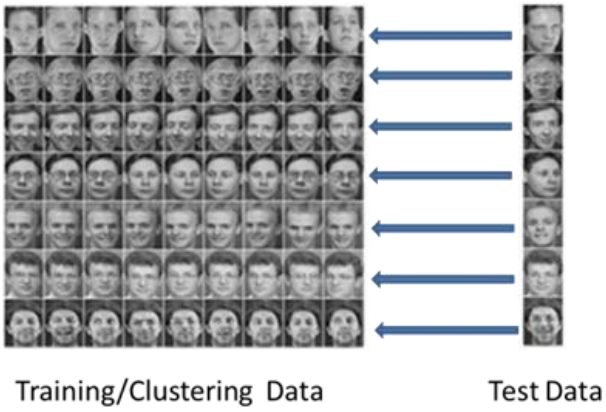


Figure 4.2 Part of ATT Data Set Partitioned for Training and Testing

This testing environment allowed us to examine two questions. First, what is the influence on performance brought by different norms used in nearest neighbor search? And second, how well our hierarchical model's performance compared with a single AM module? Each of these questions is fundamentally important to understand the capability of N-Tree. Because we have no precise model of a distance metric for the behavior of coupled nano-oscillators, and hence we try different norms to exam our model. Moreover, since the retrieval performance depends on both data set and the algorithm we use, our goal is to achieve the performance of idea single AM.

To gain some insight into the qualities of different norms, we tested different distance metrics for both the *k*-means clustering and the searching processes. The model was tested using an L2 norm, L1 norm, and L0 norm respectively.

L2 norm is the Euclidean distance as we know. On an n-dimensional vector space, the distance of two vectors x and y can be defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4-1}$$

L1 norm is also known as city block distance, or Manhattan distance, which is defined as:

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{4-2}$$

Similarly, L0 norm is defined as:

$$d(x,y) = \sum_{i=1}^{n} \delta_i \,, where\ \delta_i = \begin{cases} 1, & if\ x_i - y_i > \theta \\ 0, & otherwise \end{cases} \quad (4\text{-}3)$$

θ is a customized threshold that is determined by the data set in this case.

For comparison, between our hierarchical methods and a direct implementation of ARENA, we assume we have an ideal single large AM unit that can hold all 360 images and can classify each input image by directly comparing the degree of match to every memorized image. This model represents the best performance possible using pixel-by-pixel distance to classify the face image data.

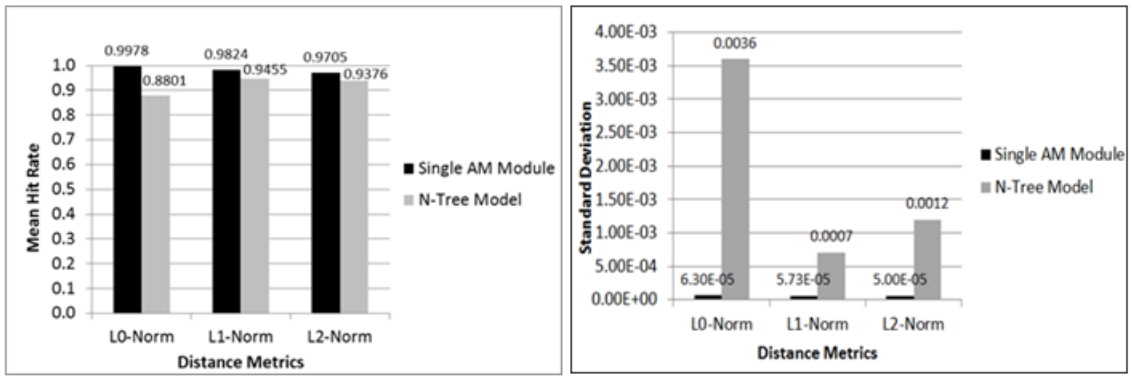


Figure 4.3 Retrieval Performance on ATT Data Set

The results of these tests are summarized in Figure 4.3. It gives the average hit rates and standard deviations of the N-Tree model and ideal single AM module under different norms. From this figure, we notice that the standard deviations of all these tests are very small compare to the average hit rates and can be ignored. The N-Tree model's variation is larger than the single AM module because different training data sets generate different trees of data clusters, which directly influence the performance of retrieval. Considering the three different norms, L0 norm is the best distance metric for nearest neighbor search, while L1 norm is the second and L2 norm is

the worst. However, L0 norm performs the worst for N-Tree model. Because it is not a good norm for k-mean clustering algorithm and thus leads to a bad data clustering.

For the ATT data set, when compared to an ideal "flat" single AM model, our model is very close to the best performance with a higher efficiency. Even though the AM performs its comparisons in parallel, for each case, the large single AM unit model has to compute the degree of match for all of the 360 stored images, while the N-Tree hierarchical model with a 3-level depth performs at most 3×16=48 operations of comparison with a time penalty of only 3x.

After verifying our model's functionality as an AM system, we employ the same noise test as we used in chapter 3 to examine the performance of the N-Tree model on a different real application. Salt and pepper noise is a form of noise typically seen on images. It appears as randomly occurring white and black pixels. We keep the training data set and test data set unchanged but make noisy input images by adding salt and pepper noise to test images with a density ranging from 0 to 0.5. Figure 4.4 shows different level of noise density.

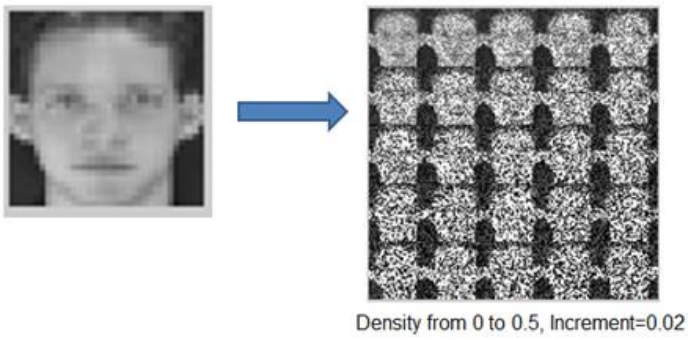


Figure 4.4 Images with Salt and Pepper Noise of Different Densities

We run the simulation for 1000 times with each image under each noise density, and count the hit rate. Figure 4.5 shows the 40 hit rate curves represent each of the 40 images' retrieval tasks. Though a noise density larger than 0.1 can make the image hard to recognize for human eyes, the N-Tree model can still achieve a high hit rate for most images.

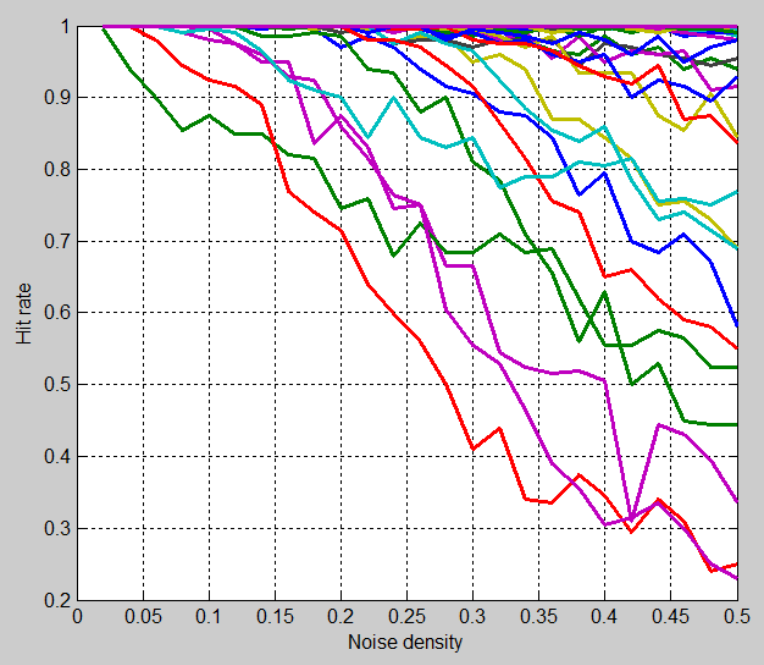


Figure 4.5 Noise Test Results of N-Tree Model

Although the tests above show good performance for the hierarchical AM model on the ATT data set, we note that our model still fails to achieve the exact performance of the large single AM model (ARENA's direct implementation). In order to test our performance on a larger and more challenging problem we used a second data set from The Facial Recognition Technology (FERET) Database [43]. The test subset we used has 2015 grayscale images of human faces from 724 people. For this data set, each subject has a different number of images, which range from 1 to 20. These images also include various facial expressions.

For the second set of tests, we use the same technique and same system specification as the ATT data set tests. That is, for each person, pick one image for testing and the rest for training. This gives 1291 images in the training data set and 724 images in the test data set. However, among these 724 subjects, the training set may have very few or even no images for some people. For example, if there is only one image for a person, this image will be used as test data and never be recognized correctly because the cluster analysis will have no instances of the

person to train with. Hence, the recognition task on this data set is more difficult than that of the ATT data set due to it having more classes, unbalanced training data, and images with diverse features.

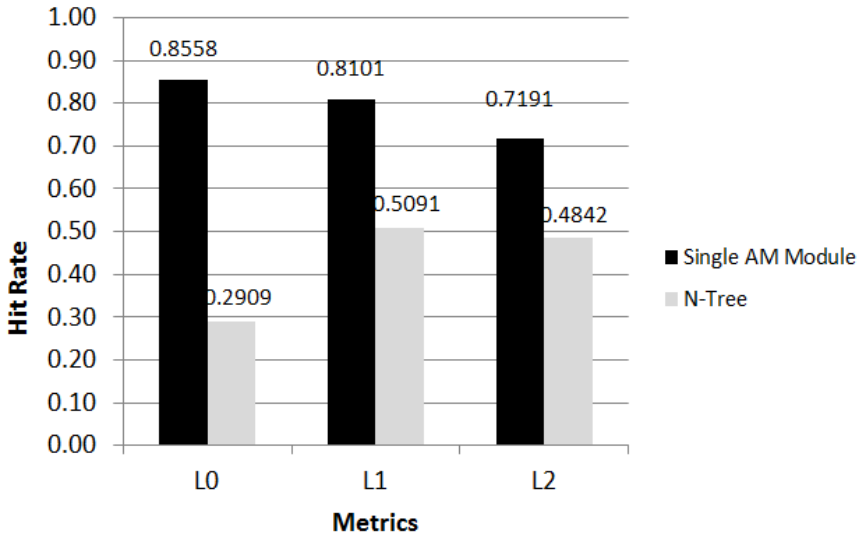


Figure 4.6 Retrieval Performances on FERET Data Set

The results are presented in Figure 4.6, shown later, which indicates that hit rates of both the single flat AM model and the N-Tree decreased when compared with ATT data set. Further, for this test data, the N-Tree model performs much worse than the single AM model with hit rates of only 29% to 51%.

4.3 FANOUT VS. PERFORMANCE

Since our goal is to obtain the same retrieval performance as a single flat AM, we explore the correlation between the clustering tree's fan-out and performance. The fan-out of the N-Tree model is the size of each AM node. For example, if the fan-out is four, each data cluster splits into four sub-clusters during hierarchical k-means clustering, and each AM node can only store

four sub-cluster centroids or image vectors. For a tree of a data set generated from hierarchical k-means clustering, fewer children for each node usually means smaller clusters, which may increase the possibility of making a mistake during the retrieval procedure. This is because once an AM unit outputs the "closest" centroid of a sub-cluster, the input pattern is compared with data in this sub-cluster only, and may miss the real closest patterns. However, larger clusters mean more comparison operations and lower speed, to give a higher possibility of finding the right answer. We repeat the same simulation on the FERET data set but vary the fan-out number from 2 to 30. The results are all plotted in Figure 4.7. As we expect, generally a higher fan-out number improves the hit rate to some extent. However, it is still far worse than our goal.

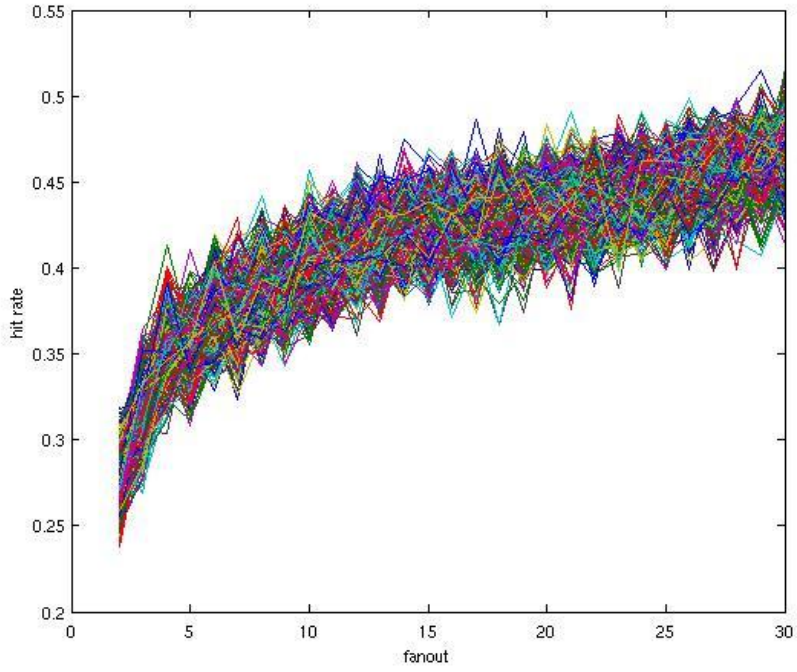


Figure 4.7 Fanout vs. Hit Rate for N-Tree model

4.4 BRANCH AND BOUND SEARCH

The simulation results above tell us that the clustered data structure generated from the hierarchical k-means algorithm cannot provide us the same performance as the flat single AM model. The reason is that the hierarchical k-means clustering is initialized randomly and might not always cluster all the data to its nearest neighbor centroid.

To improve the performance of the hierarchical architecture, we adopt an algorithm called branch and bound search [44]. The basic idea is to search additional nodes that possibly contain a better answer, after we finish a routine N-tree search. In other words, when a leaf node is visited at the end of the walk down the tree using associative processing, we backtrack on the tree and check other nodes. The branch and bound search algorithm provides a method for generating a list of other nodes that need to be searched in order to optimize the result.

In branch and bound search, a simple rule can be used to check if a node contains a possible candidate for the nearest neighbor to the input vector. For a cluster with centroid C_i, we define its radius r_i as the farthest distance between the centroid and any element in the cluster. Here we define X as the input vector and B as the distance between X and the best nearest neighbor found so far. Then, if a cluster satisfies the condition:

$$B + r_i < d(X, C_i) \tag{4-4}$$

where d is the distance function. Then no element in cluster C_i can be nearer to X than the current bound B. Initially, B is set to be $+\infty$. This rule is illustrated in Figure 4.8.

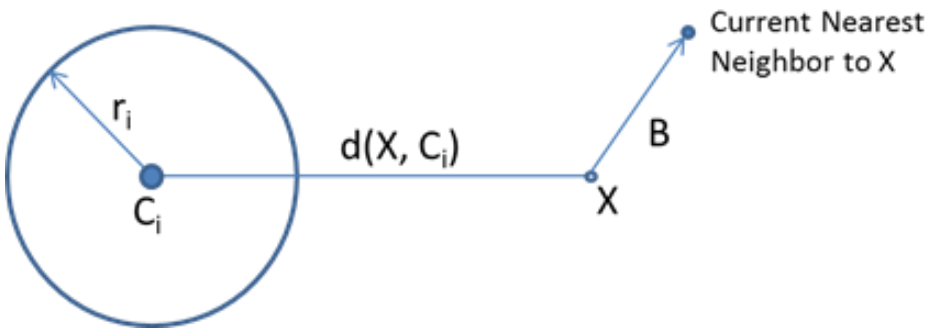


Figure 4.8 Branch and Bound Search Rule

The flow chat of branch and bound algorithm is shown as Figure 4.9. The full algorithm is summarized as:

(1) Set B = +∞, current level L=1, and current node is set to be root node

(2) Place all child nodes of root node into the active list

(3) For each node in the active list at the current level, if it satisfies the rule, remove it from the active list.

(4) If the active list at the current level is empty, backtrack to the previous level, set L=L-1, terminate algorithm if L=0, go to (2) if L is not 0. If there is node in the list, go to (4)

(5) In the active list at the current level, choose the node that has the centroid nearest to input vector and call it the current node. If the current level is the final level, go to (5). Otherwise set L=L+1, go to (1)

(6) In the final level, choose the stored vector which is nearest to the input vector X, and update B. Back track to previous level, set L=L-1, go to (2)

Theoretically, branch and bound search can be guaranteed to find the nearest neighbor of an input vector from a hierarchical k-means clustered data set, which means it should have the same recognition performance as the flat single AM model.

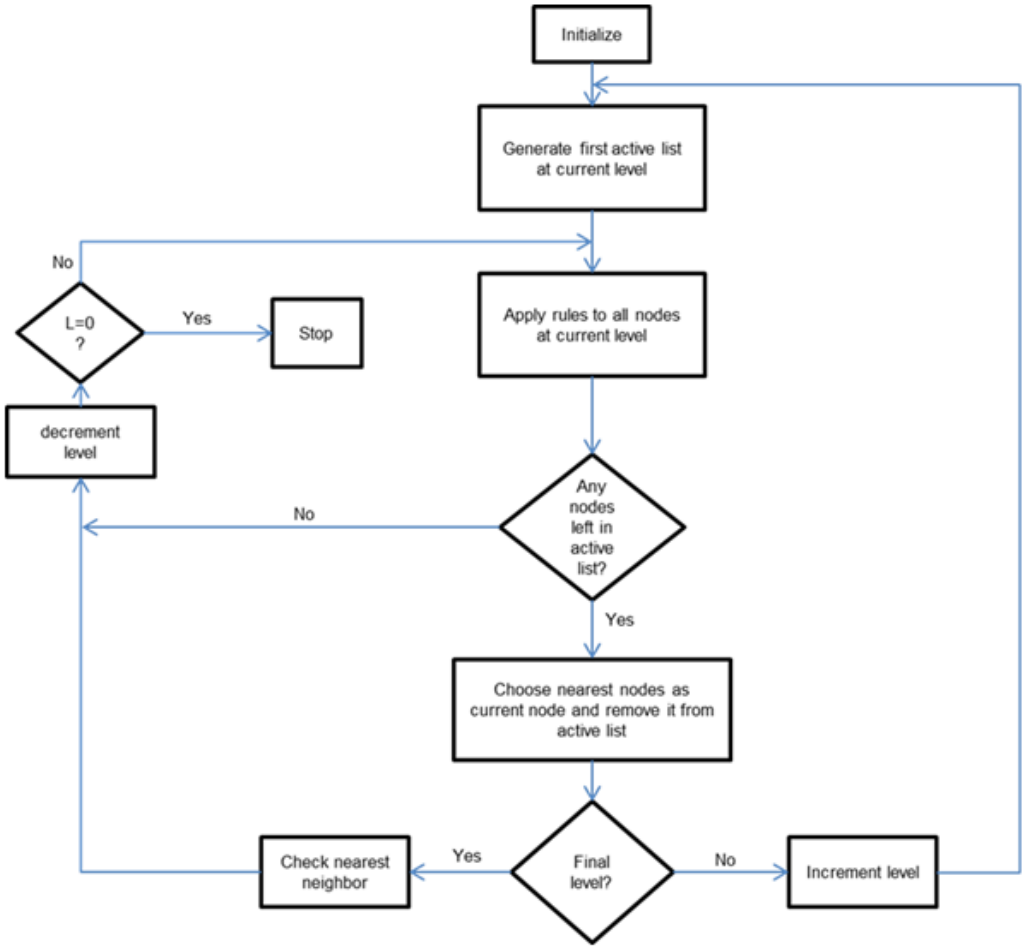


Figure 4.9 Flow Chart of Branch and Bound Search

Therefore, we enhance the N-Tree AM search with the branch and bound search algorithm and repeat the face recognition task. With the same training model, we see that branch and bound search always achieves the exact same performance as the ARENA algorithm, regardless of data set and fanout for the N-Tree model. Figure 4.10 compares the hit rates of the flat (ARENA) model, the pure N-Tree model, and the branch and bound algorithms when the fanout is four. The results indicate that the N-Tree model with branch and bound search can achieve the same performance of the ideal single AM module.

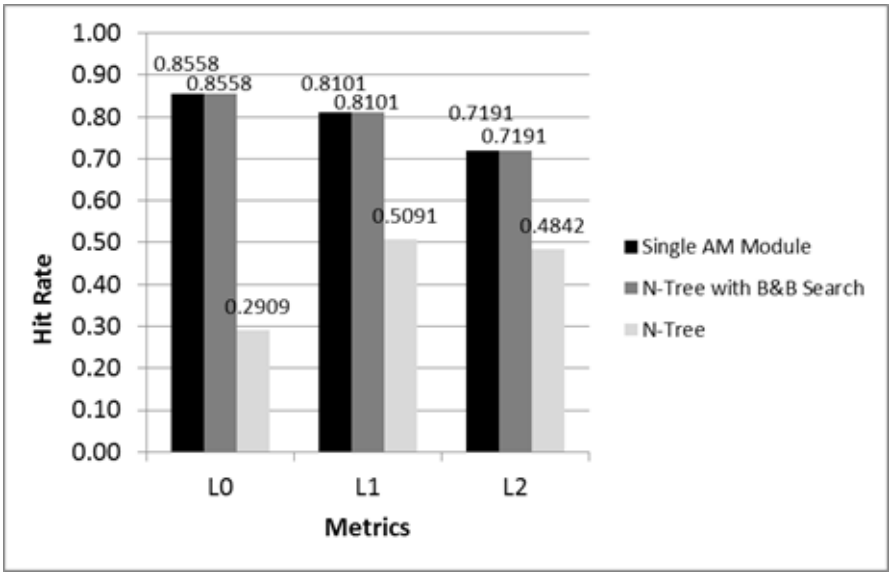


Figure 4.10 Retrieval Performances on FERET Data Set

Table 4.1 gives us the number of nodes visited for the branch and bound search, and indicates the efficiency of this technique compared to the simple N-Tree search with L1 norm. We can notice that the price of higher performance is lower efficiency. Without branch and bound search, the number of nodes visited is always just the depth of N-Tree structure, on average 9.3. To test all the possible candidates for nearest neighbor, the algorithm has to access additional nodes in the tree, on average 159.9 with a standard deviation of 86.2. This extra work is what is required to achieve the comparable performance of the flat network shown in Figure 4.10. Thus, there is an obvious tradeoff between hit rate and processing speed. The original search routine for the N-Tree model is "depth-only" one-way from the root node to a leaf node. In other words, if one AM unit in the middle level makes a wrong decision due to an imprecise clustered data set, the final output vector will be incorrect. On the contrary, branch and bound search provides opportunities to backtrack and search more possible clusters in other AM units with the cost of search speed. As Table 4.1 shows, the number of nodes that branch and bound search visits is 17 times as many as the original search of N-Tree model.

Table 4.1 Nodes Visited In Branch and Bound Search

B&B Search	Total Nodes	Visited Nodes	Visited Leaf Nodes
Max	1199	454	293
Min	795	9	1
Mean	956.7	159.9	59.6
Standard Deviation	34.3	86.28	47.4
Average N-Tree Size	**Total Nodes**	**Depth**	**Leaf Nodes**
	956.7	9.3	846.7

To speed up the search, we adjust the branch and bound search by fixing the condition of checking rule as follows:

$$B + \alpha \cdot r_i < d(X, C_i) \tag{4-5}$$

where α is a factor used to reduce the effective radius of the clusters, and ranged from 0 to 1. For some clusters, most elements are far away from X, only very few of them are in the bound. These clusters rarely contain the nearest neighbor but by the previous rule, to guarantee optimality, we have to search them. Reducing the cluster's radius can help us avoid these less likely clusters and access many fewer nodes in the process.

In a final test, we sweep the value of α from 1 to 0, and observe the trends of hit rates and number of nodes visited during the search. The result is shown in Figure 4.11. We notice that the slopes of the three curves are different. Nodes searched decreases as radius reduced. But the hit rate stays the same until α gets 0.4. By reducing the cluster radius to some extent, we can improve the efficiency of searching without sacrificing the performance.

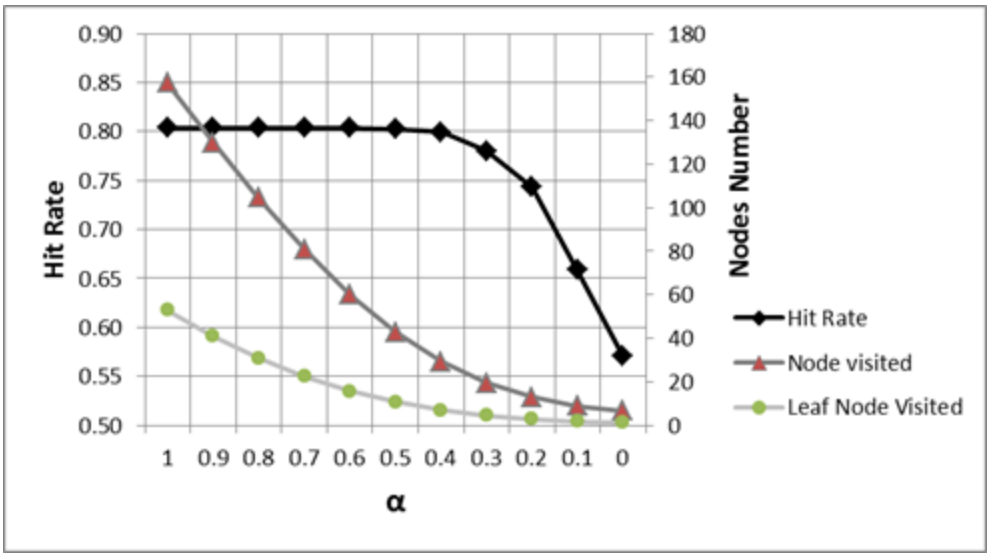


Figure 4.11 Performance vs. Speed for Modified Branch and Bound Search

4.5 CONCLUSION

In this Chapter, we propose our second hierarchical model, the N-Tree model, as an oscillator based AM system. The purpose is to deal with pattern recognition tasks for large data sets in the real number field and for high dimension spaces. First, we introduce our method to organize the data set by clustering data into a tree structure. Then, we describe the architecture of the N-Tree model. AM nodes in the N-Tree only store a representation of the data clusters. For the training of the N-Tree model we use hierarchical k-means clustering. Then we use two human face data sets ATT and FERET for model simulation. On the ATT data set, the N-Tree model provides very good performance for both the function test and noise test, close to the ideal single AM unit. However, it does not work well on the more challenging FERET data set. We analyze the reason of this and introduce a branch and bound search algorithm and successfully improve the

performance. Finally, we modified the branch and bound search algorithm and discuss the tradeoff of speed and performance in the retrieval process of the N-Tree model.

5.0 CONCLUSIONS AND FUTURE WORK

We have presented two hierarchical architecture models of associative memory to address the pattern recognition problem in high dimension vector spaces. Our exploration of AM models indicates that it is possible to use an Oscillatory Neural Network as a basic AM unit built from nano-oscillators. Give the limits and demerits of current device technology, we build our AM system hierarchically and design algorithms to improve the performance of hierarchical AM systems.

In the rest of this chapter, we summarize the work in this thesis again. Then we list the main contributions of our work and the important conclusions we can draw from experiments. Finally, we introduce some remaining problems in this area and a few of our ideas to extend our work.

5.1 SUMMARY

We first introduce the basic concept and brief history of associative memory in Chapter 2. Then in the rest of this chapter we focus on three AM models related to our work – the Palm Network, the Hopfield Network, and the oscillatory neural network. These models are analyzed and their characteristics discussed in detail. We also establish and simulate a Matlab model of the Hopfield Network, and the oscillatory neural network. Their functionalities and relationships are

demonstrated through simulation results. Then we build a hardware-based model of the oscillatory neural network with PLLs and simulate this model. In addition, alternative devices are proposed and possible methods of implementation are explored.

Chapter 3 describes our first hierarchical AM model. First we present the motivation to build the AM architecture hierarchically and design a simple prototype containing three levels of AM units. We use the Palm network with a unary code representation pattern as the AM model, which is simple to simulate and explore. We explore this prototype by training it with four 8 by 8 square binary images and compare it with an ideal fully connected single flat AM. To solve the problems such as pattern conflict, capacity limits and information abstraction, we provide an improved version this hierarchical model. A set of binary images that are converted from JPEG pictures are employed for tests of our new model and the results from different tests are shown for analysis. Finally, we explore dynamic on-line learning as a new possible function of our model.

In Chapter 4 we propose a different hierarchical model called the N-Tree model. We first show how the data in this model is clustered and represented. And then based on this data structure, a corresponding hierarchical tree structure is designed for AM units. The training and retrieval processes of the N-Tree model are described through an example of a small size tree structure. Two data sets for human face recognition, ATT and FERET, are used to test the functionality and retrieval performance of the N-Tree model. We discuss the simulation results and the branch and bound search algorithm to improve retrieval performance. Finally, we modified the search rules of this algorithm to balance the speed and performance.

5.2 CONTRIBUTIONS

The contribution that our work makes to the area of pattern recognition and the application of emerging nano-device technology can be summarized as the following:

- The approaches of implementing oscillatory neural networks on nano-oscillators as a basic associative memory model are explored and discussed. Due to the high frequency of nano-oscillators, this model can execute the nearest pattern matching with a very high speed.
- As the compensation of difficulties in hardware implementation of large size AMs, a hierarchical AM model is proposed and improved. This model can partition vectors in high dimension spaces into short vectors and process them with multiple layers of AM units, and also approach the performance of an ideal large single AM.
- Dynamic online learning of the hierarchical AM model is explored, which demonstrates that the system can change the prior probability based on repeatedly training and has potential in advanced inference.
- The N-Tree model is proposed to address the capacity problem of the previous model. It can be trained on large data sets in the real number field. The N-Tree model is designed based on the architecture of AM unit built from nano-oscillators.
- Branch and bound search algorithm is introduced and modified to improve the retrieval performance of the N-Tree model without greatly impacting search speed.

5.3 CONCLUSIONS

Based on the exploration of this work, a set of important conclusion can be drawn:

- Associative memory is a distributed memory that outputs the "closest" pattern to the input pattern. The degree of match is determined by its internal dynamics. For oscillatory neural networks, patterns are stored in the attractor basins of the system's energy function, formed by adjusting the coupling strength weights between oscillators. In the retrieval process, the output of the network converges to a nearest attractor from an initial state.
- Oscillatory neural network can be implemented on a cluster of coupled nano-oscillators. The degree of match between any two patterns can be obtained from these oscillator clusters and they can be utilized to build an AM unit.
- The first hierarchical AM model we proposed can retrieve patterns in a high dimension vector space, but it has problems like small capacity, pattern conflict, and information abstraction. However, the improved model exhibits strong resistance to noise of image patterns and is capable of retrieving heavily distorted patterns. It also reduces the pattern conflict in the retrieval process and provides a solution to the information abstraction problem. The capacity is determined by the size of AM unit in the top level, which limits the scalability of this model.
- Dynamical online learning makes the hierarchical model capable of retrieving patterns depending on prior probabilities. It can be realized by adding a multiplicative factor on degree of match or varying the coupling coefficient of oscillators.

- The N-Tree model, serves as an AM system for high dimension real number vectors, can approach the performance of an ideal model of nearest neighbor search according to our simulation based on three common distance metrics. We cannot guarantee the same results for a real model built by nano-oscillators because the degree of match varies for different devices and this technology dependent behavior remains unknown.
- The retrieval performance of the N-Tree depends on the distance metric we use and also the data base. These two factors influence the quality of hierarchical k-means clustering.
- Branch and bound search algorithm improves the N-Tree model's hit rate by increasing the number of search operations. It enlarges the search area and thus increases the possibility of finding the nearest pattern. In another words, it compensates for the negative effect brought by k-mean clustering algorithm. The upper bound of hit rate is the idea single large AM which compare all stored patterns with input pattern.
- Branch and bound search algorithms slow down the retrieval process of the N-Tree model by increasing search operations as we mentioned. This tradeoff can be optimized by varying the search rule of branch and bound search. By using the L2 norm, we can make the N-Tree model achieve the ideal performance while increasing the search time by only a factor of two.
- Theoretically branch and bound search can only work by distance norms that satisfy the triangle inequality. The effect of nonlinear distance metrics from the real of oscillators needs more research.

5.4 FUTURE WORK

Our work presented in this thesis is just some initial exploration of this new topic. Future work can be summarized by the following points:

- From the device and circuits level, the detailed method of coupling the nano-oscillator and supporting circuits to obtain the degree of match still needs more exploration. For example, some nano-oscillators can be coupled electrically or magnetically, or controlled by voltage/current feedback. Moreover, the degree of match, represented by the convergence quality, can be tested through convergence speed, stability of output signal, or time-frequency analysis. There exist many options to implement the oscillatory network model.
- From the view of a mathematic model, the distance metric in the nearest neighbor search of the AM model still remains unknown as the dynamics of weakly coupled nonlinear oscillators are very complex, and this needs more mathematic analysis.
- From the architecture level, we might try to merge the two models proposed in this thesis so that we can take the advantage both of them. The first model exhibits more potential in advanced functionality like inference and invariant recognition that we did not explore in the current research, while the second model is more scalable, practical and flexible for different data sets.
- From algorithm level, more research in the search of tree structure can help us improve the speed and performance of our model. Another problem is how to actually realize algorithms, like branch and bound, effectively in hardware. On the other hand, we can

develop new algorithms based on our model for feature extraction, segmentation, edge-detection.

- We may need to develop an emulator so that we can explore more characteristics of our hierarchical models. It can also serve as a CAD tool for future design.

BIBLIOGRAPHY

[1] A.W. and A.R. Burks, The ENIAC: The First General-Purpose Electronic Computer, Annals of the History of Computing, Vol. 3 (No. 4), 1981, pp. 310–389

[2] John von Neumann, First Draft of a Report on the EDVAC, 1945

[3] D.A. Patterson, J.L. Hennessy, and D. Goldberg, Computer Architecture: A Quantitative Approach. San Francisco: Morgan Kaufmann, 1996.

[4] Hakin, S. Neural Networks: A Comprehensive Foundation, Prentice Hall, New Jersey, 1999.

[5] R.H. Dennard, F.H. Gaensslen, H.N. Yu, V. L. Rideout, E. Bassous and A. R. LeBlanc, "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions," IEEE J. Solid-State Circuits, SC-9, pp. 256, 1974.

[6] S.H. Lo, D.A. Buchanan, Y. Taur and W. Wang, "Quantum-Mechanical Modeling of Electron Tunneling Current from the Inversion Layer of Ultra-Thin-Oxide nMOSFET's," IEEE Electron Device Lett, Vol. 18, pp. 209 –211, 1997.

[7] G.I. Bourianoff, J.E. Brewer, R.K. Cavin, J.A. Hutchby and V.V. Zhirnov, "Boolean Logic and Alternative Information-Processing Devices", Computer, Vol. 41, Issue 5, pp. 38- 46, May 2008

[8] S.I. Kiselev, et al; "Microwave Oscillations of a Nanomagnet Driven by a Spin-polarized Current", Nature 425, pp.380-383, Sep. 2003

[9] F.C. Hoppensteadt, E.M. Izhikevich, Weakly Connected Neural Networks. New York: Springer-Verlag, 1997.

[10] F. Corinto, A. Ascoli and M. Gilli, "Nonlinear Dynamics of Memristor Oscillators", IEEE Transactions on Circuits and Systems, Vol. 58, No. 6, Jun. 2011

[11] D. Weinstein and S.A. Bhave, "The Resonant Body Transistor", Nano Lett., 2010, 10(4), pp 1234–1237

[12] M.H. Devoret and C. Glattli, "Single-electron Transistors", Physics World, pp.29-34, Sep. 1998

[13] L. Lapique. Sur L'excitation Electrique Des Nerfs, Journal of Physiology, pp.620-635, 1907

[14] C. Mead. Analog VLSI and Neural Systems, Addison-Wesley, Reading, MA, 1989

[15] W.S. McCulloch and W.Pitts, "A Logical Caculus of Ideas Immanet in Nervous Activity", Bulletin of Mathmatical Biophysics, 5, 1943

[16] A.L. Hodgkin and A.F. Huxley, "Currents Carried by Sodium and Potassium Ions through the Membrane of Giant Squid Axon of Loligo", Journal of Physiology, 116:449-472, 1952

[17] J. Hawkins and G. Dileep, Hierarchical Temporal Memory: Concepts, Theory and Terminology. Numenta Inc. (2006). http://www.numenta.com.

[18] M. Riesenhuber and T. Poggio, "Hierarchical Models of Object Recognition in Cortex", Nature Neuroscience, 2, pp.1019-1025, 1999.

[19] F.C. Hoppensteadt and E.M. Izhikevich, "Oscillatory Neurocomputers with Dynamic Connectivity," Phys. Rev. Lett., vol. 82, pp. 2983–2986, 1999.

[20] K. Steinbuch and U.A.W. Piske, "Learning Matrices and Their Applications", IEEE Transactions on Electronic Computers EC-12: pp: 846-862,1963

[21] D.J. Willshaw, O.P. Buneman, et al, "Non-Holographic Associative Memory." Nature 222: pp. 960-962, 1969.

[22] J. Buckingham and D. Willshaw, "Performance Characteristics of the Associative Net," Network: Computation in Neural Systems 3: 407-414, 1992.

[23] R.N.A. Henson and D. Willshaw, "Short-Term Associative Memory", Proceedings of the INNS World Congress on Neural Networks, 1995, Washington D.C.

[24] J.A. Anderson, J.W. Silverstein, et al. "Distinctive Features, Categorical, Perception, and Probability Learning: Some Applications of a Neural Model", Psychological Review 84: 413-451, 1977

[25] J.A. Anderson. The BSB Model: A Simple Nonlinear Auto-associative Network. Associative Neural Memories: Theory and Implementation, M. H. Hassoun, Oxford University Press, 1993

[26] J.A. Anderson, P. Allopenna, et al. "Programming a Parallel Computer: The Ersatz Brain Project", Studies in Computational Intelligence, Springer Berlin / Heidelberg, 2007.

[27] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceedings of the National Academy of Sciences, Vol. 79, No.8, pp. 2554-2558, 1982.

[28] G. Palm, "On Associative Memory," Biological Cybernetics, Vol. 36, pp. 19-31, 1980

[29] A.J. Storkey and R. Valabregue, "The Basins of Attraction of a New Hopfield Learning Rule," Neural Networks Vol. 12, No. 6, pp. 869-876, 1999

[30] F.C Hoppensteadt and E.M. Izhikevich, "Synaptic Organizations and Dynamical Properties of Weakly Connected Neural Oscillators: I. Analysis of Canonical Model," Biological Cybernetics, Vol. 75 pp. 129-135, 1996

[31] Shibata, Tadashi, "Neuron MOS Binary-logic Integrated Circuits: I. Design Fundamentals and Soft-hardware-logic Circuit Implementation," IEEE Transaction on Electron Devices Vol.40, No. 3, pp. 570-576. 1993

[32] M. Tsoi, et al, "Excitation of a Magnetic Multilayer by an Electric Current," Physical Review Letters, Vol. 80, pp. 4281-4284, 1998

[33] M.R. Pufall, W.H. Rippard, S. Kaka, et al., "Frequency Modulation of Spin-transfer Oscillators," Applied Physics Letters, Vol. 86, No.8, pp. 082506 – 082506-3, 2005.

[34] W.H. Rippard, M.R. Pufall, S. Kaka et al., "Injection Locking and Phase Control of Spin Transfer Nano-oscillators," Physical Review Letters, Vol. 95, No.6, pp. 067203, 2005

[35] D. George. "How the Brain Might Work: A Hierarchical and Temporal Model for Learning and Recognition," Electrical Engineering, Stanford University, 2008.

[36] Zhu, Shaojuan, "Associative Memory as a Bayesian Building Block," Computer Science and Electrical Engineering, Oregon Health & Science University, 2008

[37] Personal communication with Denver Dash, Intel Science & Technology Center on Embedded Computing at Carnegie Mellon University

[38] S.P. Levitan, Y. Fang, "Non-Boolean Associative Architectures Based on Nano-oscillators," 13th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), 2012

[39] R. O. Duda, P. E. Hart, and Stork, D. G. Pattern Classification, John Wiley & Sons: 2000.

[40] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[41] T. Sim, R. Sukthankar, M.D. Mullin, and S. Baluja, "High-Performance Memorybased Face Recognition for Visitor Identification," Proceedings of IEEE Conference on Automatic Face and Gesture Recognition, Grenoble, pp. 214-220, 2000.

[42] P.A. Devijver, and J. Kittler; Pattern Recognition: A Statistical Approach, Prentice-Hall, London, GB, 1982

[43] http://www.itl.nist.gov/iad/humanid/feret/feret_master.html

[44] K. Fukunaga. “A Branch and Bound Algorithm for Computing k-Nearest Neighbors,” IEEE Transactions on Computers, Vol. C-24, No. 7, pp. 750-753, Jul 1975