6195 92

QA
76
.Z3
Z8

A BIDIRECTIONAL ASSOCIATIVE MEMORY BASED ON CORTICAL SPIKING NEURONS USING TEMPORAL CODING

by

Masood Zamani

Bsc, Amir Kabir University, Iran, 1998

A thesis

Presented to Ryerson University

In partial fulfillments of the

Requirements for the degree of

Master of Science

In the program of

Computer Science

Toronto, Ontario, Canada, 2009

A BIDIRECTIONAL ASSOCIATIVE MEMORY BASED ON CORTICAL SPIKING NEURONS USING TEMPORAL CODING

Masood Zamani

Msc, Computer Science, Ryerson University, 2009

ABSTRACT

In this thesis, we proposed a spiking bidirectional associative memory (BAM) using temporal coding. The information processing in biological neurons is beyond of that applied in the current Artificial Neural Networks (ANNs). The coding scheme used in ANNs known as "mean firing rate" cannot answer the fast and complex computations occurring in the cortex. In biological neural networks the information is coded and processed based on the timing of action potentials. To improve the biological plausibility of the standard BAM, we employed spiking neurons for its processing units, and information is presented to the BAM in the form of temporal coding. The neurons employed in the model are heterogeneous, and being able to generate various spike-timing patterns. Genetic Algorithm and Co-evolution are used for training, and the experimental results of the proposed BAM are compared to those of the standard BAM. The results show improvements in recall, storage capacity and convergence which are of interest to design a BAM.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Alireza Sadeghian who was abundantly helpful and offered invaluable assistance, support and guidance. I gratefully acknowledge Dr. Sylvan Chartier for his guidance and contribution.

I would also like to convey thanks to the members of the supervisory committee Dr. Isaac Woungang, Dr. Marcus Santos, and Dr. Tim McInerney.

TABLE OF CONTENTS

LIST OF TABELS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The pioneering work of exploring the neural system was marked by Ramon Cajal who discovered the primary structure of the neural systems. Afterward neurophysiologists and many scientists in related disciplines have explored the structure and operation of the neural system in detail. The neural system is the most advanced and complex structure responsible for the central processing and controlling hub in the human and other species.

One area of neuroscience research that has attracted significant attention is the computational neuroscience that explores the information contents of neural activities by modeling the neural systems mainly through two approaches. The first approach is to model biological neurons and its biophysical compartments. The second approach is to model a network of interconnected neurons that represent the part of a complex neural system, and then finding the answers to the fundamental questions related to the neural coding and information processing in the brain. Investigating different models of biological neurons, neural networks and simulating the models by fast computational resources are the examples of neuroscience research that substitute the conventional methods of understanding the information processing in neural systems.

In addition, computer scientists are trying to solve number of scientific and engineering problems by inspiration from biology. Artificial Neural Networks [1] and Evolutionary Computation [2] have become the two dominant fields of study in computational intelligence. These methods are applied in many scientific and engineering problems such as pattern recognition, control, optimization and data mining.

1.2 The information processing and action potentials

How the brain processes the information and what kind of coding scheme used in this complex structure are the questions that have inspired many scientists in multi-disciplined fields to find the answers. The classic experiments of Adrian known as "The basis of sensation" have revealed three fundamental and universal features of neural coding and become the key steps to understand how the outside world is represented to the brain by firing time of neurons [3]. These neural principles are:

1. Each sensory neuron, such as muscle and motor neurons, produces an stereotyped action potential or spike. This is known as "all-or-none" law. After receiving stimuli these cells either produce or do not produce action potentials which are propagated along the cell's axon to the other neurons. In other words, there is no other form of intermediate signals, and consequently the set of arrival times of spikes called "spike train", generated by sensory neurons, provide information to the brain. These elementary units of neural coding are called "action potentials".

2. The second neural coding principle is that by increasing a stimulus, such as a continuous load on a stretch receptor, rates of action potentials are increased as well. This phenomenon, known as "rate coding", can measure the intensity of a stimulus by counting the number of spikes in a defined time interval. This means the spike rate is a function of stimulus intensity with specific parameters that are chosen by neurons known as "feature selectivity". For instance, in visual neurons these parameters can be an object luminance, size, shape and color.

3. Adaptation is the third important exploration of Adrian that shows if a static stimulus is presented during a lengthy period of time, the rates of action potentials decrease comparing to the time that the stimulus is presented to a cell [4].

Another important issue about the action potentials in a neural system is the randomness of the neural responses. If the "all-or-none" law experiment is repeated by the same stimulus, we end up with several spike trains that are not identical. One primary solution is to calculate the average of spikes over time. However if the trail is not possible to be repeated, the characteristics of a neural response can't be quantified.

The relation between the spike train and real world signals (stimuli) represented to the sensory neurons is an essential concept to understand the neural coding in neural systems. More precisely, by creating a set of rules we will be able to understand the meaning of the neural responses which are in the form of spike trains corresponding to the applied stimuli.

In the real world there may be several stimuli that simultaneously are received by sensory neurons and as explained previously there is not one to one relation. To formulize the variability, conditional probability theory is applied such that the probability of observing a spike train $\{t_i\}$ given the stimulus $s(t)$ is calculated. Then, the reverse relation which is the likelihood of observing the stimulus $s(t)$ given the spike train $\{t_i\}$ is calculate by Bayes' theorem [5]. Entropy has also been used to quantify the relation between spike trains and stimulus. The entropy of a neuron's spike train on the presence of a stimulus over a period of time is computed. If the same trail is repeated, however there would be a different spike train and as well as a different computed entropy. The difference between the two entropies gives the information that a spike train provide about the stimulus [6].

In addition to revealing the neural codes by understanding the relation between a stimulus and a spike train, there are several neural coding schemes regarding the amount of information carried by the timing of spikes. In general the coding schemes are categorized to the Rate coding and Spike coding which are discussed in Chapter 2.

1.3 Spiking neurons

In addition of exploring the information processing in biological neural system, this is important to investigate what kind of artificial neuron model is best suited to represent its counterpart biological neuron's properties. There are several spiking neuron models for simulating the dynamics of cortical neurons. In a general, these models fall in three categories.

The first group are the artificial spiking neuron models whose dynamics are described by a number of differential equations. These models consist of the parameters explicitly representing their counterpart biological neurophysiologic components such as ionic currents. Hodgkin-Huxley model [7] is an example of spiking neuron model in this group. This model describes the dynamics of a biological neuron with its known components. However, it is difficult to set the accurate values for its parameters experimentally. The model consists of sixteen differential equations and ten variables. Therefore, simulating a large neural network that employs the spiking neuron is computationally expensive. A simplified model of Hodgkin-Huxley was proposed by [8]. However, the simplified model does not have the capabilities of the Hodgkin-Huxley model in terms of generating all known neuro-computational spiking patterns which are believed to have significant roles in spike-timing information processing.

The second group of spiking neuron models are also represented with several differential equations. However, they do not include the parameters corresponding to physical properties of a biological neuron explicitly. The membrane potential of a neuron is the element that describes the dynamics of the neurons. For instance, Resonate-and-Fire spiking neuron [9] and Izhikevich's spiking neuron model [10] belong to the second category of cortical spiking neuron models.

The third group of spiking neuron models are the threshold-fire neurons that describe the neuron's computation by integrating inputs and firing at a defined threshold. Integrate-and-fire (I&F) and Spike Response Model (SRM) are two examples of this group. I&F parameters are depended on the voltages of neurons whereas in SRM the parameters depend on the time of the last spikes. One issue in I&F model is that the neurons act as integrators which is not always the case in reality because real neurons exhibit spiking patterns such as resonance, inhibition, and etc [11]. The descriptions related to the most common spiking neuron models are provided in Chapter 2.

The computational cost and biological likelihood of a spiking neuron model are very important to evaluate the neuron model. The range of neuro-computational patterns that a spiking neuron model supports measures its biological likelihood. Neuro-computational patterns, or spiking patterns, are the dynamical features that biological neuron are able to generate at the presence of stimuli. There are over twenty known spiking patterns such as tonic spiking and phase spiking. Hodgkin-Huxley spiking neuron model is the most complete model among the twelve common spiking neuron models. It demonstrates better ability to support all dynamical features of a biological neuron and its biophysical correlates. However, its computational cost is the highest among all models. On the other hand, Integrate-and-fire model support least neural features but its computational cost is the least.

Therefore, by considering the two criteria among the spiking neuron models, Izhikevich's spiking neuron model proposed recently is the best model. The model is as biologically plausible as Hodgkin-Huxley model and computationally efficient as Integrate-and-fire model. The dominance of the model based on the two criteria has been investigated in [12] through comprehensive comparisons among the other common spiking neuron models.

1.4 Spiking Neural Networks

Artificial neural networks (ANNs) have been successfully applied to many cognitive and engineering problems. ANNs have been improved gradually since its first introduction. The primary models were able to compute digital information [13].One such that improvement was the use of activation function for their processing units. It enables them to compute continuous values and generating outputs of specific ranges (or normalized firing rate). Although this improvement was significant, and it increased the computational power of artificial neural networks, however there are number of shortcomings as the following:

- The fast and complex computations in the biological neurons in the cortex area cannot be explained by the neurons used in ANNs. The dynamics of biological are very complex. Biological neurons exhibit over 20 neuro-computational patterns such as phase spiking, tonic bursting, phase bursting, spike latency, and etc. The ability of generating these patterns is highly important in neural computation context.
- The features and properties of biological neuron cannot be integrated and tested in the current artificial neuron models.

Nonetheless, artificial neural networks that use activation functions their normalized outputs (or normalized firing rates) can be interpreted as "mean-firing rate" coding. This coding is similar to the operational mode of biological neurons in high cortical areas known to fire in various frequencies between their minimum and maximum frequencies [14]. However, with regards to the fast analog computation of biological neurons in the cortex, "mean-firing rate" scheme is a debate in research community. For instance, humans are able to analyze and classify visual patterns within 100 ms. The process involves 10 synaptic stages from retina to the temporal lobe. It has been shown that a single cortical area involved in visual processing can complete its process in 20-30 ms. On the other hand the firing rates of neurons involved in this computation are usually below 100 Hz and at least 20-30 ms is needed only to sample the firing rate of a neuron [15].

Many experimental results from neurobiology have led to the theory that biological neural systems use the timing of action potentials or spikes to encode information [4]. The results from the biological experiments in years have inspired the investigation of a different type of neural networks known as spiking neural networks (SNNs) which apply spiking neurons for their computational units.

Unlike the current artificial neural networks, SNNs are able to expressing spatial-temporal properties of biological neurons [16]. These properties are not considered by common neural networks. An important characteristic of spiking neural networks is that they are naturally embedded in time. Spike latencies, axonal delays, refractory periods, neuron resonance and network oscillations provide the ability to process time-varying data in a more natural and computationally powerful way than is available by the second generation models [17].

To build a spiking neuron network, there are issues such as processing time, training and biological likelihood of the spiking neurons that are embedded to the network. Very realistic spiking neuron models are computational expensive and small number of them used in a simulation takes huge amount of time, and it is even impossible to simulate a large network. In addition, many well-known learning methods such as Hebbian and Back-propagation learning methods [18] have to be adapted for spiking neural networks due to their complex spatial-temporal structure and processing unit models. The learning methods of SNNs are bounded by their spiking neuron models and structures. In other words, they cannot be generalized for all spiking neural networks. For instance, SpikProp [19] was proposed only for the simplified spike-response neuron model by adding several restrictions and assumptions. ReSuMe learning method also is applied with a specific structure that employs a different spiking neuron model [20].

It has been shown that the neural networks which employ continuous real-valued activation functions can be constructed by spiking neural networks using a delay coding named temporal or firing order coding [21]. The spiking neuron model employed in the model is the simplified version of spike-response model. The study has shown theoretically that a feed-forward spiking neural network with temporal coding can emulate any arbitrary feed-forward neural networks and the model is several times faster than the current ANNs, implemented via frequency-coding or firing rates, and the spiking neural network requires less number of neurons.

Hopfield has also simulated radial base function by spiking neurons with temporal coding [22]. The model does not provide a learning rule for adjusting weights, and a multi-layer neural network cannot be simulated.

The spiking neural network model proposed by [21] also has been applied for modeling a spiking bidirectional associative memory proposed in [23]. The complexity of the spiking BAM's structure is higher than the standard BAM's since it uses a multilayer neural network. In addition the spiking neuron model employed in the model is the simplest one, thus it does not take advantage of the advanced spiking neuron models.

Moreover, a model of SNN architecture has been also proposed to construct a non-linear function approximator, and the model is able to simulate any real valued function [24]. This network employs leaky integrate-and-fire spiking neuron model. In addition, the learning method used in the network is based on gradient descent method, unlike the model used in [21] which applies Hebb's rules.

It has been shown that spiking neural networks can also be used to model auto-associative memory (Hopfield network) with temporal coding. The structure of the model is similar to the Hopfield network and its processing units are based on spike-response model [25].

1.5 Objective

Knowing the weaknesses of the current artificial networks and the advantages of spiking neural networks, in this thesis we aim to construct a different type of bidirectional associative memory (BAM) that employs an advanced cortical spiking neuron model. Three distinguishing characteristics of the proposed spiking BAM compared to the standard BAM are:

1. The likelihood of its functional units (neurons) to the biological neurons.

2. Processing of information in the form of action potentials.

3. Representing information by the temporal coding scheme.

In other words, by designing the spiking bidirectional associative memory, we aim to construct a more realistic BAM whose processing units operate the way as biological neurons. In addition, the information presented to the spiking BAM is in the form of action potentials in various firing times. Presenting information in this form is similar to the information provided to the biological neural system by sensory neurons. After implementing the spiking BAM, we will conduct a number of experiments to investigate the model performance versus similar experiments performed on the standard BAM.

1.6 Methodology

To achieve the implementation of the proposed spiking bidirectional associative memory described in 1.5, we outline the four main parts of the methodology that will be described in detail in Chapter 3. These main parts are:

1. The structure of the proposed model is similar to the standard BAM which is a two-layer recurrent neural network.
2. The processing units of the proposed neural network are based on the Izhikevich's spiking neuron model.
3. The coding scheme used for the model is the temporal coding scheme which is a variation of the spike coding scheme.
4. The proposed model is trained by Genetic Algorithm and Co-evolution methods.

1.7 Results

In this research we conducted several experiments to investigate training, recall, storage capacity, convergence and the number of iterations needed to recall patterns. These are the important factors to design a successful model of bidirectional associative memory. The results show number of improvements in recall, storage capacity and convergence. Moreover, the biological plausibility of BAM's processing units has been increased, and the model is able to process information that is presented in the form of firing times.

1.8 Thesis outline

In order to accomplish the study, Chapter 2 focuses on background related to current approach on associative memory, spiking neuron models, neural coding schemes, spiking neural networks, and spike-timing encoding methods. In Chapter 3, the proposed spiking BAM is described. Chapter 4 provides the experimental setup, the results, and discussion about the model performances in the several aspects. Chapter 5 presents the summary and conclusion, and future research and the application of the model are pointed out.

CHAPTER 2: BACKGROUND

2.1 Associative memory

Associative memory is the intrinsic and primary function of the human brain. When we see an acquaintance our brain maps the pattern of the face to the person's name, or it associates a name to a phone number. Associative memory may also be used as part of a system. For instance, in banking system the password that a client enters is mapped to the image of an original password. The computer memory also associates an address to data. The other type of associative memory is called Content-Addressable Memory (CAM) which associates data to the address of other data. This type of memory is also known as Hopfield memory. Hopfield is the pioneer of modeling associative memory by artificial neural networks [26].

Essentially, to model an associative memory we need to represent the patterns in the form of *n*-dimensional vectors. Therefore, there might be *k* pairs of patterns (vectors) such as

$$\{(X_1,Y_1),\ldots,(X_i,Y_i)\} \qquad i=1,\ldots,k \tag{2.1}$$

and a function $\psi(X)$ that maps each X_i to Y_i . The dimension of vectors (patterns) X_i, Y_i may be different, and the mapping function can be a linear or non-linear. The vectors are in the Hamming space which means their components are either +1 or -1. In Hamming space, the number of mismatches between the corresponding components of two vectors represents the Hamming distance of the two vectors. The following sections describe the mathematical representation of associative memory which is in three categories [27].

2.1.1 Hetero-associative memory

In this type of association, the mapping function $\psi(X)$ is modeled as

$$\psi(X_i)=Y_i \qquad i=1,\ldots,k \text{ (number of patterns)} \qquad (2.2)$$

and if a pattern X with the closest Hamming distance to X_i is represented to ψ, then $\psi(X)$ must be equal to Y_i such that $\psi(X)=Y_i$.

2.1.2 Interpolative associative memory

This is similar to the hetero-associative memory, however, if a given noisy pattern X differs from X_i by the vector α then the mapping function is presented as

$$\psi(X_i+\alpha)=Y_i+\beta=Y \qquad i=1,\ldots,k \qquad (2.3)$$

Pattern Y differs from Y_i by the vector β. If we try to minimize the distance of Y_i and β by an optimization methods then ψ becomes a hetero-associative memory.

2.1.3 Auto-associative memory

In this type of associative memory each pattern is mapped to itself thus the mapping function ψ is constructed such that

$$\psi(X_i)=X_i \qquad i=1,\ldots,k \qquad (2.4)$$

and if a given pattern X has the closest distance to X_i then $\psi(X)=X_i$.

If the patterns are unit vectors and orthogonal, the mapping function ψ can be implemented such as

$$\psi(X)=(Y_1.X_1^{\mathrm{T}}+Y_2.X_2^{\mathrm{T}}+\ldots+Y_k.X_k^{\mathrm{T}}).X \qquad i=1,\ldots,k. \qquad (2.5)$$

For instance, for k orthogonal patterns based on hetero-associative function ψ we have

$$\psi(X_3)=(Y_1.X_1^{\mathrm{T}}+Y_2.X_2^{\mathrm{T}}+\ldots+Y_k.X_k^{\mathrm{T}}).X_3=Y_1.X_1^{\mathrm{T}}.X_3+Y_2.X_2^{\mathrm{T}}.X_3+Y_3.X_3^{\mathrm{T}}.X_3+Y_k.X_k^{\mathrm{T}}.X_3=Y_3$$

where $X_i . X_j = 1$ for $\forall\ i, j$ that $i=j$, and $X_i . X_j = 0$ for $\forall\ i, j$ that $i \neq j$.

In case a set of patterns are not orthogonal the mapping function $\psi(X_i + \alpha) = Y_i + \beta$ is not linear and the error can be calculated such as

$$\beta = (Y_1 . X_1^T + Y_2 . X_2^T + \ldots + Y_k . X_k^T) . \alpha \qquad i=1,\ldots,k. \qquad (2.6)$$

2.2 Bidirectional associative memory

Bidirectional associative memory (BAM) is a dynamical system that belongs to the family of hetero-associative memory [28] as shown in Figure 2.1. BAM is a fully connected recurrent neural network that has two layers. These layers can act as both inputs and outputs. The information from the first layer (X) is propagated to the second layer (Y) and the output of the second layer is propagated to the first layer, and one such cycle is referred to as a reverberation (or iteration). BAM is an extended version of Hopfield network that associates two different patterns with different dimensions.

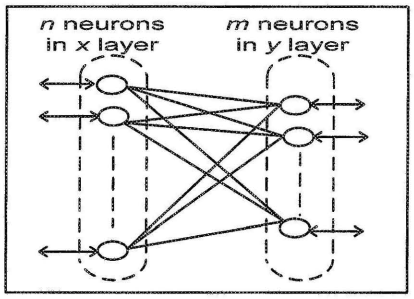


Figure 2.1 The schematic view of a standard BAM

The values of the connection weights are calculated as

$$w = x_1 . y_1^T + x_2 . y_2^T + \ldots + x_i . y_i^T \qquad i=1,\ldots, p \text{ (number of patterns)} \qquad (2.7)$$

The equation (2.7) is based on the Hebbian learning. The mapping function to calculate the outputs in both X and Y layers are

$$Y=W.X \qquad \& \qquad X=W^T.Y \tag{2.8}$$

2.2.1 Processing information in BAM

Recalling stored patterns and noise corrections are an iterative process in BAM network. Let's assume that A_1, B_1 are the noisy version of A, B which are the two stored associated patterns to each other. At the beginning, A_1, B_1 are represented to the second and first layer respectively and the outputs are generated as follows

$$b_{i+1}^{K} = \begin{cases} +1 & \sum_{j=1}^{n} \mathrm{w}_{kj}.a_j > 0 \\ b_i^k & \sum_{j=1}^{n} \mathrm{w}_{kj}.a_j = 0 \\ -1 & \sum_{j=1}^{n} \mathrm{w}_{kj}.a_j < 0 \end{cases} \quad a_{i+1}^{q} = \begin{cases} +1 & \sum_{j=1}^{m} \mathrm{w}_{qj}^{T}.b_j > 0 \\ a_i^q & \sum_{j=1}^{m} w_{qj}^{T}.b_j = 0 \\ -1 & \sum_{j=1}^{m} w_{qj}^{T}.b_j < 0 \end{cases} \quad k=1,\ldots,m \;\; \& \; q=1,\ldots,n \tag{2.9}$$

$b^k{}_{i+1}$ is the updated output of component k from the second layer Y and $a^q{}_{i+1}$ is the updated value of the component q from the first layer X. n and m are the numbers of neurons used in X, Y layers respectively. After updating the outputs of all nodes in both layers, the new vectors A_2,B_2 are sent back to the Y and X layers. The cycle of updating the outputs and feeding back to the opposite layers is repeated until there is not further changes between the vectors A_i, A_{i+1} and B_i, B_{i+1} at the iterations i and i+1, and A_{i+1}, B_{i+1} are the fixed points.

2.2.2 BAM energy function

In the BAM, the weight matrix is calculated by (2.7), and afterward the weights are the fix part of the system. However, the recalling process is an iterative and dynamical process. When the noisy patterns A_1, B_1 are presented to the network, they are changed based on equation (2.9) until the networks reaches a stable point. Thus, there would not be further changes to the outputs. These vectors are changed as a function of time and it makes BAM to be a dynamical system.

In a dynamical system, to show that there is a stable state, its energy function must be a Lyapunov function. Based on the Lyapunov's theorem, a dynamical system has a stable state if it meets the following conditions [27]:

- Any change in the system must result in decreasing the energy level.
- There must be a boundary to the minimum level of energy.
- The time to reach a stable point or minimum energy level must not be infinite. In other words, the changes of energy level must not be extremely small leading to reach minimum energy in infinity. BAM energy function is defined as

$$E(X,Y) = -Y^T.W.X \quad or \quad E(X,Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} y_i.w_{ij}x_j \tag{2.10}$$

The energy function E meets the criteria set by Lyapunov's theorem. Therefore, BAM is a dynamical system that has a stable state according to the following:

- Changing the vectors presented in the X, Y layers by updating the outputs in each step, decrease the energy level.
- There is a minimum boundary for energy value which is equal to

$$E_{\min} = -\sum_{i=1}^{m}\sum_{j=1}^{n}\left|w_{ij}\right| \tag{2.11}$$

- A stable state is reached in finite time.

Based on the above criteria, the energy level has a minimum boundary. As a result, it cannot move to the negative infinity and the stable state is reached in finite iterations. The following is the mathematical explanation that shows the decrease of energy during recall process [27]. Let assume that by propagating the vector X to the second layer Y the component q of this layer is changed. Therefore y_q is changed to y'_q. According to (2.10)

$$E(X,Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} y_i.w_{ij}x_j = -\sum_{i=1}^{m} y_q.w_{ij}x_j - \sum_{\substack{i=1 \\ i=q}}^{m}\sum_{j=1}^{n} y_i.w_{ij}x_j \tag{2.12}$$

$$E'(X,Y) = -\sum_{i=1}^{m} y'_q.w_{ij}x_j - \sum_{\substack{i=1 \\ i=q}}^{m}\sum_{j=1}^{n} y_i.w_{ij}x_j \tag{2.13}$$

By subtracting the two energy value (2.12) and (2.13) , the difference is equal to

$$\Delta E(x,y) = E'(x,y) - E(x,y) = (y_q - y'_q)\sum_{i=1}^{m} w_{qi}x_i \tag{2.14}$$

If y_q is changed from +1 to -1 then based on (2.9) the second part of (2.14) is negative and the first part of the equation (2.14) is positive, thus ΔE is negative. In case y_q is changed from -1 to +1 according to the equation (2.9) the second part of the equation (2.14) must be positive and the first part becomes negative. Consequently, the multiplication is negative. As a result, in both cases the energy values are decreased. The procedure can be generalized when more than one component of Y layer is changed.

2.2.3 BAM storage capacity

The number of patterns that can be stored and recalled correctly is limited in BAM. If the network is over loaded by patterns, a phenomenon known as crosstalk occurs. It means by representing a noisy version of one of the stored pattern, the network instead of converging toward a stored pattern, converges to a pattern that is not similar to any stored pattern. It has been shown by Kosko [28] that the maximum number of patterns, *p*, is equal to

$$p = \text{Min}\,(m, n) \qquad n = X\text{'s dimensions},\ m = Y\text{'s dimensions} \tag{2.15}$$

There is another conservative evaluation that heuristically shows the real maximum capacity is equal to

$$p = \sqrt{Min(n, m)} \tag{2.16}$$

2.3 Genetic Algorithm

Genetic Algorithm (GA) [2] is a heuristic optimization method which effectively used in Machine Learning. It has successfully been applied to various problems related to theoretical sciences and engineering. GA was inspired by Darwin's theory of "the survival of the fittest". The strength of the method is in its exploitation and exploration properties. In other words, this method not only attempts to find out the best solution in its local search but also searches for other alternative and possibly better solutions in the entire area of the search space. The complexity and magnitude of search or solution space varies from one problem to another one. In GA, primary or candidate solutions for a given problem are represented in a fixed length of strings or more precisely chromosomes. These coded strings are called genotype. Before converting an initial solution to the form of chromosome representation, the solution is called phenotype. A chromosome consists of genes which are the components of a solution such as parameters. Alleles or bits also are the components of a gene and their values, for example in binary representation, are 0 and 1. In GA, there must be a mechanism to evaluate each individual's fitness or the quality of solutions. The methods of fitness evaluation are varied, and it depends to the type of problem being solved.

GA uses genetic operators inspired from natural random selection and recombination. These operators are applied to the individuals of a population and the offspring with the higher fitness are passed to the next generation. This evolving process is continued until the desired solution is reached.

The start point in GA is to create an initial population. A population consists of individuals that are the primary solutions. These primary solutions are the initial states in the solution space. A solution space can be considered as a graph that we can move from one node (solution or state) to the others. In GA there are number of evolutionary-inspired operators and they basically act as shifting the individuals of a population to different states. These operators are mutation, crossover, reproduction and selection.

2.3.1 Mutation

Mutation transforms the values of selected alleles within a gene. In binary representations shown in Figure 2.2 one is flipped to zero and vice versa. Although each genetic operator has its own advantages, mutation is an important operator that maintaining diversity in population.

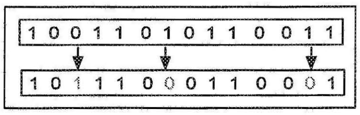


Figure 2.2 Mutation in alleles

2.3.2 Crossover

Crossover operator is applied on two individuals. One or two points are selected within the range of a chromosome length. The selected parts of the chromosomes are exchanged between the two individuals as shown in Figures 2.3, 2.4.

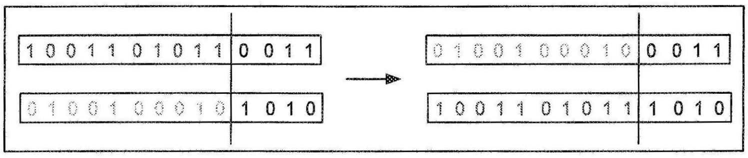


Figure 2.3 One-point Crossover

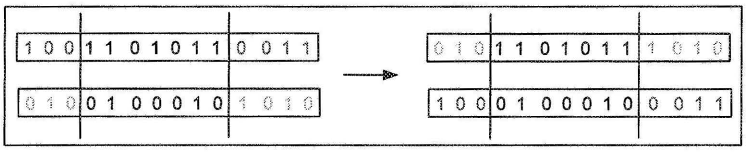


Figure 2.4 Two-point Crossover

2.3.3 Reproduction

Reproduction operator is the simplest one. The selected individual is duplicated and directly passed to the next generation. Usually a few best individuals of a generation are selected since they can contribute their good traits to the next population.

2.3.4 Selection

The selection operator is applied to the entire population. The best individuals are selected and collected in a selection (candidate) pool. The genetic operators are applied to the selected individuals in the pool. There are several methods of selection. However two common selection methods are Fitness-proportional selection and tournament selection. The following are a brief explanation of the methods [29]:

- In tournament selection a group of individuals with the size of k, tournament size is randomly selected. In this group individuals compete with each other and the individuals with best traits, winners, are mutated and passed to the selection pool. The rest of the individual, losers, are returned to the population. This routine continues until an specific number of individuals are passed to the selection pool.

- In Fitness-proportional selection method a probability is assigned to each individual in the population. These probabilities determine the chance that an individual can pass their offspring to the next generation. Obviously, the individuals with higher probability are selected more. The probability of i^{th} individual in a population of size n is calculated as

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \qquad f_i = \text{The fitness value of } i^{th} \text{ individual} \qquad (2.17)$$

2.3.5 The genetic algorithm steps

To apply the Genetic Algorithm the following steps are taken:

1. Representing the primary solutions in a fix length of strings (chromosomes) by a proper coding scheme.
2. Finding a suitable fitness function for evaluating each individual (solution).
3. Defining the rates that each type of the genetic operator is chosen during evolution.
4. Defining the maximum number of generations that the evolution cycle should continue.
5. Creating n random solutions in the initial population.
6. Creating a candidate pool from the recent population.
7. Applying the genetic operators on all individuals of the candidate pool.

8. Evaluating the offspring by fitness function and passing the winners to the next generation.
9. If the termination criteria are not met repeating the steps from 6 to 9 on the recently created population. The termination criteria are met when either the desired solution is obtained or the maximum number of generations is reached.

2.4 Co-evolution

Co-evolution is an important phenomenon for the evolutionary process in nature where two or more different species compete with each other and affect each other evolutionary processes. For example, plants over time evolving to make stronger exterior to prevent insect eating them. On the other hand, insects also at the same time evolving stronger jaws enabling them to eat those plants and this competitive process may continue over time for any changes on the both species.

Regarding to the above explanation the principles of co-evolution have been adopted and modeled in evolutionary computations to increase its performances for complex problems. By this model, an original problem including several components is decomposed to several sub-problems and for each one a separate population and consequently evolutionary process takes place. To obtain and evaluate a desired solution for the whole system, the selected sub-solutions from each population are integrated and evaluated as one complete solution for the system [30].

2.5 Biological neurons

A biological neuron consists of three main functional parts that are called dendrites, soma and axon. The dendrites are a neuron's input devices. The soma is the central processing part of the neuron, which performs non-linear processing. The axon carries the neuron output signal. The junction between two neurons is called synapse. In the nervous system shown in Figure 2.5 the neuron sending its output signal to the other neurons is named pre-synaptic neuron and the neuron that receives the signal is called postsynaptic neuron.

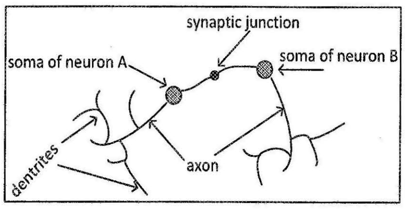


Figure 2.5 A schematic view of two connected biological neurons [11]

The neural signals are short electrical pulses called action potentials or "spikes" with the amplitudes of approximately 100mv and the duration of 1-2ms. Spikes are the wave of electrochemical activity. The form of the pulses or spikes depicted in Figure 2.6 does not change while propagated along the axon.

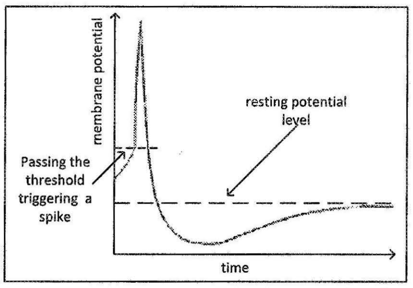


Figure 2.6 The form of an action potential (spike) in a biological neuron [11]

In a sequence of pulses or "spike train" since the form of pulses are similar, they do not carry any information. However, the timing of spikes in a spike train has meaning and carrying information. In a spike train, the timing of spikes may or may not be at regular intervals. After a neuron sends a spike, it is impossible for the neuron to generate the second spike immediately even the neuron receiving exceeding input signals from the pre-synaptic neurons. The period of time that is impossible for a neuron to fire a spike is called "absolute refractory time". The reason is that the threshold value is increases to infinitive and over time decreases until it reaches the minimum level as shown in Figure 2.7. After this period of time, within a certain time it is hard to exit neuron to generate spike ,however It is not impossible and this period of time is called relative refractory time.

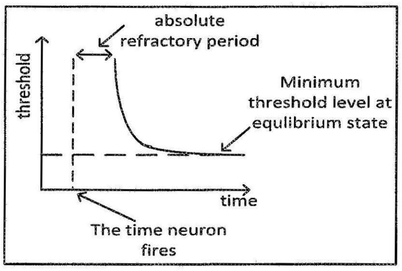

absolute refractory period
threshold
Minimum threshold level at equlibrium state
time
The time neuron fires

Figure 2.7 The form of threshold at the time of spike and after that in a biological neuron [16]

In case a neuron does not receive input signals, the neuron‘s membrane potential won't be changed and the neuron is at rest or equilibrium state. The membrane potential $u(t)$ is the potential differences between the interior of the cell and its surroundings. Resting potential is about –65mv with negative polarization. Depolarization of $u(t)$ happens when an input reduces the negative polarization of membrane potential through an excitatory synapse and hyper-polarization happens when an inhibitory synapse increases the membrane negative potential [31]. In a simple mathematical expression a spike denoted $t^{(f)}$ is generated at time t when the neuron's potential $u(t)$ passes the threshold θ from below or

$$\frac{du(t)}{dt} > 0 \ \& \ \ t=t^{(f)} \tag{2.18}$$

The effect of a spike fired by a pre-synaptic neuron on the potential of post-synaptic neuron is called post-synaptic potential (PSP) shown in Figure 2.8. PSP on excitatory neuron denoted as EPSP and for inhibitory neuron IPSP.

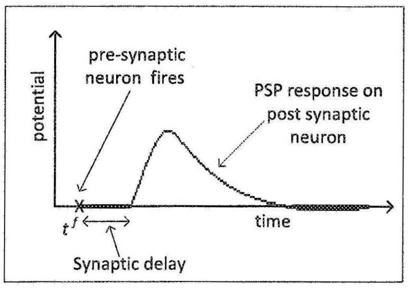

potential
pre-synaptic neuron fires
PSP response on post synaptic neuron
t^f
time
Synaptic delay

Figure 2.8 The form of PSP at the time of spike and after that in a biological neuron [16]

2.6 Spiking neuron models

To describe the dynamics of biological neurons several spiking neuron models have been proposed, and in a broad category the models can be grouped in two as the following:

2.6.1 Hodgkin-Huxley

Hodgkin-Huxley's spiking neuron model is the most complete spiking neuron model that not only is rich on generating all known spiking patterns but also includes all known biological correlations [7]. This model describes that the semi-permeable cell membrane which separates the interior of cell from the extracellular liquid reveals dynamical properties similar to those of a capacitor and mathematically can be shown as

$$I_{total}(t) = I_c(t) + \sum_k I_k(t) \tag{2.19}$$

$$c\frac{du}{dt} = -\sum_k I_k(t) + I_c(t) \tag{2.20}$$

The sum of ionic currents is $\sum_k I_k(t)$, and $I_c(t)$ is the external current applied to cell. The membrane voltage is denoted $u(t)$. This model has three channels that are opened and closed at different times and lets the ions flow in or out of cell and change the equilibrium state of the neuron .The two ionic channels are K^+ ,Na^+ and the third is a leakage channel for unknown current. There are several simplified versions of Hodgkin-Huxley model which the most common one is Leaky Integrated-and-Fire model. It models the dynamic of a neuron by an RC circuit.

2.6.2 Spike Response Model (SRM)

Spike Response Model (SRM) [32] is the simplest spiking neuron model that represents the dynamic of a neuron at time t by the firing times of all pre-synaptic neurons sending spikes to a post-synaptic neuron *i* up to time *t*, unlike Hodgkin-Huxley spiking neuron model that the dynamic of a neuron at time t described by a set of differential equations as a RC circuit. The mathematical form of the SRM model is described as

$$u_i(t) = \eta(t-\hat{t}_i) + \sum_{j=1}^{n} w_{ij} \sum_{f} \varepsilon_{ij}(t-t_j^f) + u_i(0) \tag{2.21}$$

The equation (2.21) describes the potential of a neuron *i* at time *t*. The *j* neurons (*j*=1,..., *n*) are the pre-synaptic neurons connected to neuron *i* through the synaptic junctions. W_{ij} is the synaptic efficacy or weight for the link from the neuron *j* to the neuron *i*. The last firing time of the neuron *i* denoted as $\hat{t}_i$ and the function η describes the potential changes of neuron *i* at the time $\hat{t}_i$ of spike and afterward. The function ε which is called response function describes the post synaptic potential (PSP). PSP is the effect of pre-synaptic potential on the potential of neuron at destination. In other word, let assume neuron *i* is in equilibrium state at time t_0 and only neuron *j* sends one spike to neuron *i* at time *t*. Thus, the PSP at the destination is equal to

$$\varepsilon(t-t_0)=u_i(t)-u_i(t_0) \tag{2.22}$$

In formal SRM model the function ε is replaced by Dirac delta function, therefore the summation $\sum_f \varepsilon_{ij}(t - t_j^f)$ is, in fact, over the firing times or spike train (2.23) of neuron j.

$$\{t^1_j, t^2_j, \ldots, t^f_j\} \tag{2.23}$$

The dynamic of threshold function and PSPs in two simplified models [16] of the SRM are shown in Figures 2.9 and 2.10.

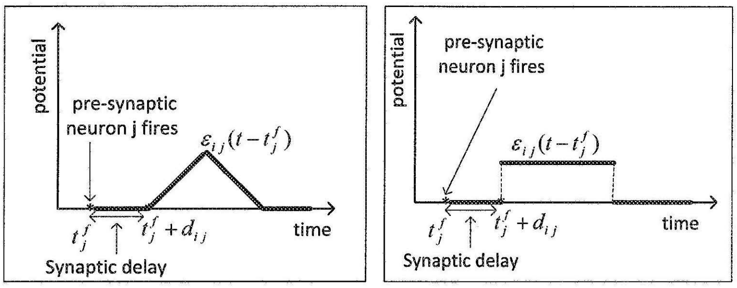

potential
pre-synaptic neuron j fires
$\varepsilon_{ij}(t-t_j^f)$
t_j^f
$t_j^f + d_{ij}$
time
Synaptic delay
potential
pre-synaptic neuron j fires
$\varepsilon_{ij}(t-t_j^f)$
t_j^f
$t_j^f + d_{ij}$
time
Synaptic delay

Figure 2.9 The dynamics of PSP function in two simplified SRM spiking neurons [16]

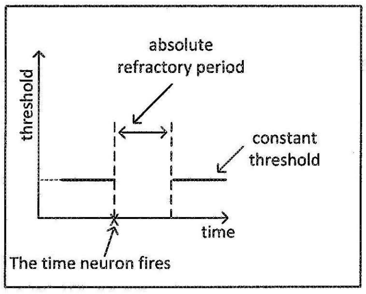


Figure 2.10 The dynamics of threshold function in the simplified SRM [16]

2.7 Neural coding

In general to investigate the problem of neural coding, we should find answers for the following questions:

- The patterns of spikes contain what type information.

- The neurons employ what form of coding and decoding schemes to transmit and receive the information.

More importantly, we want to read the neural patterns and understand their meanings. However, there are number of speculations and there are not complete answers for these questions. It's been studied that in the mammalian brain more than 10^{10} neurons exist. In such a complex structure thousands spikes are fired only in a fraction of a second.

Basically, there are two theories to tackle the neural coding problem. The first and conventional scheme is the Rate code and the second is the Spike code. In the following we discuss these two points of view.

2.7.1 Rate code

By this coding scheme the number of pulses (spikes) is counted within a defined time interval, usually 500ms, and a temporal average is calculated [4]. This method is known as "mean firing rate" and the idea is that the information is coded in temporal averages. There are three different approaches for the rate coding method.

1. Using spike count method, the number of spikes generated by a single neuron is counted within a time interval on a single trail. In a trail, a stimulus is represented to a neuron, and its response which is a spike train is examined. The temporal average v is obtained by dividing the number of spike by the time interval

$$v = \frac{n_s}{T} \qquad T = \text{time interval} \qquad (2.24)$$

2. Spike density method is similar to the spike count, however temporal average is calculate over several trail by a single neuron as

$$v = \frac{1}{T}.\frac{n_s}{K} \qquad n_s = \text{the number of spikes in } K \text{ trails} \tag{2.25}$$

3. Based on the population activity method the temporal average is calculated over a group of n neurons functioning similarly and a single trail is performed during a time interval T and the temporal average calculated as

$$v = \frac{1}{T}.\frac{\sum_{j=1}^{n}\sum_{f}\delta(t-t_j^f)}{n} \tag{2.26}$$

where $\sum_{f}\delta(t-t_j^f)$ is the spike train of neuron j and the neuron response function ε is replaced with δ function.

2.7.2 Spike code

This coding is not based on the strategies that calculating the temporal average as the Rate code methods. Instead spike coding [33] considering the timing of each single spike within a relatively very shorter time interval, about 10 ms, compared to the time interval of the Rate code. The following lists different schemes of spike code and comparing the amount of information each can carry according to the information theory.

1. Count coding: Count coding is the weakest type of spike code and it is similar to spike count. However, it does not calculate the average of spikes over time T and instead

simply counting the number of spikes. The amount of information is carried by n neurons is equal to $\log_2^{(n+1)}$ bits of information.

2. Binary coding: In binary code the firing of a neuron in time interval is considered one ,otherwise it is considered 0.Then, these values are put together as a string of 0 and 1 and interpreted as a binary number. This coding carry $\log_2^{2^n}$ bits of information by n neurons.

3. Temporal coding: Temporal coding is the most powerful scheme that the exact timing of spikes for neurons is considered. For instance, t_1 is the firing time of the first neuron , t_2 for the second neuron and t_n for the neuron n ($0 \leq t_1, t_2, \ldots, t_n \leq T$).The amount of information over time T with n neurons is equal to $n.\log_2^T$.

4. Rank order coding: In rank order code, a rank number is assigned to each neuron according to its firing time. For instance, in a 10ms time interval a neuron fires at 4ms obtaining rank 4 and rank 9 is assigned to the other neuron firing at 9ms and so on. The amount of information for n neurons in T ms is equal to $\log_2^{n!}$.

5. Synchronization coding: By synchronization coding the neurons that firing approximately in the same time are grouped together. For example with 4 neurons, if the first neuron fired at early 4ms and the second neuron 2 fired at late 2ms, these two neurons are considered in group A. On the other hand, if third neuron fired at late 7ms and fourth neuron fired at early 10ms, these two neurons are put in group B. Having a population of n neurons and g groups within time T, the amount of information is equal to $\log_2^{(g+1)^T}$ bits.

The amount of information (number of bits) that each spike code scheme can carry when n=15, T=15 and g=4 is shown in Table 2.1.

Table 2.1 The amounts of information that each coding schemes carries

Spike count	Binary code	Temporal code	Rank order code	Synchronization code
4 bits	15 bits	58.6 bits	40.2 bits	34.8 bits

2.8 The properties of spiking neuron model

To choose a cortical spiking neuron model for constructing a spiking neural network, the following factors are important to be considered:

- The model richness in terms of generating neuro-computational features of the biological neuron.
- Its computational costs and complexity.

It has been studied that neuro-computational features contribute significant roles in spike-timing information processing and especially in temporal coding. Neuro-computational features in other words are the firing patterns generated in form of spikes by biological neurons [12].

There are about twenty types of firing patterns such as tonic spiking, phase spiking, and spike latency and so on. These firing patterns are recorded by attaching an electrode to the neuron and measuring its membrane potential changes while a DC current as a input is applied to the neuron that is in equilibrium state. For example to describe the dynamics of spike-timing patterns, we explain Tonic spiking. This pattern is produced by a class of neuron called regular spiking neuron (RC).It is one type of excitatory cortical neuron that firing spikes continuously as long as the input dc-current injected to the neuron is on.

The second aspect to choose a spiking neuron model is the computational cost. The computational cost of simulating a few connected neurons does not have any impact on the system. However if a network consists of thousands neurons, it has huge computational effect, and it may not be even possible to simulate such a network. Moreover, using a powerful computational resource is not the realistic solution because the neural processing speed in human brain is less than 100 Hz. But, it can computes very complex tasks such as pattern recognition in a fraction of second.

2.9 Izhikevich's spiking neuron model

Izhikevich's spiking neuron model [10] is the only model that produces all known neuro-computational patterns same as Hodgkin-Huxley model which is the most complete biological spiking neuron model. It has also the least computational cost similar to integrate-and-fire model which produces the least neuro-computational patterns among the other models [12].The comparison of Izhikevich's model with the other common spiking neuron models for its computational cost and biological plausibility are shown in Figure 2.11.

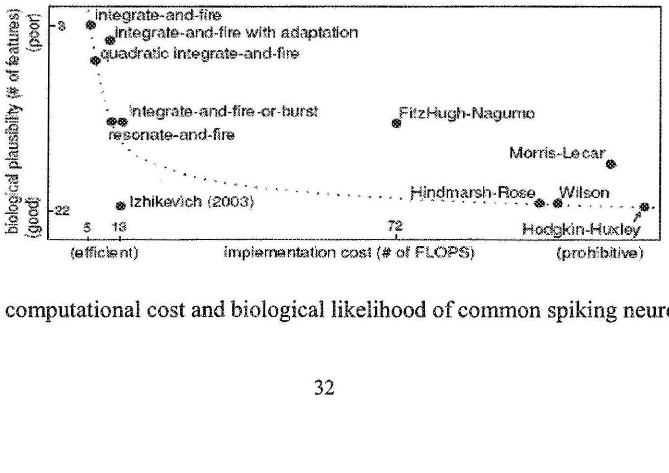


Figure 2.11 The computational cost and biological likelihood of common spiking neuron models [12]

Izhikevich's spiking neuron model is based on mathematical model and consists of two differential equations and four parameters *a, b, c, d* as follows

$$\frac{dv(t)}{dt} = 0.04v^2(t) + 5v(t) + 140 - u(t) + I(t) \tag{2.27}$$

$$\frac{du(t)}{dt} = a(bv(t) - u(t)) \tag{2.28}$$

The variable *v* represents the neuron's membrane potential. The variable *u* describes the dynamics of the membrane potential *v* for recovery after the neuron fired a spike. The parameter *a* describes the time scale of the recovery variable *u*. By choosing smaller values, the recovery becomes slower. The sensitivity of the recovery variable *u* to the threshold changes of the membrane potential *v* is defined by the parameter *b*. The level of the dynamic threshold decreases by setting greater values for the *b* parameter. The *c, d* parameters are for resetting the membrane potential *v* and recovery value *u* respectively after an action potential occurred.

After a neuron fires a spike, $v \geq +30$ mV, first *v* is reset to +30, and in the next step the value of the parameter *c* is assigned to *v*. Meanwhile, the value of *u* is changed by adding the parameter *d* to *u* as follows

$$\textit{if } \mathrm{v} \geq +30 \text{ then } \begin{cases} \mathrm{v} = \mathrm{c} \\ \mathrm{u} = \mathrm{u} + \mathrm{d} \end{cases} \tag{2.29}$$

By resetting the membrane potential to +30, the apexes of all spikes are equalized to a unique value. The threshold value in this model is ranging from -70 to -50 and it depends to the state of the neuron at time t .This means when the membrane potential reaches the mentioned rage, it fires. In the equations (2.27), (2.28) u represents the membrane recovery variable which corresponds to the activation of k^+ ionic currents and inactivation of Na^+ ionic currents.

Let consider λ is the time step in ms scale for the changes of the variables u, v. To update the values of the membrane potential v and membrane recovery u from time t to the time $t+\lambda$, a fix-step first-order Euler method is used such that if a dynamical system is represented as

$$y' = f(x) \tag{2.30}$$

then

$$y(t+\lambda) = y(t) + \lambda.f(y(t)) \tag{2.31}$$

Therefore, according to (2.31) the value of v, u at time $t+\lambda$ are computed as

$$v(t+\lambda) = v(t) + \lambda.(0.04v^2(t) + 5v(t) + 140 - u(t) + I(t)) \tag{2.32}$$

$$u(t+\lambda) = u(t) + \lambda(a(bv(t) - u(t))) \tag{2.33}$$

The dynamics of v and u over time before and after an action potential and the effects of the parameters a, b, c, d on the variables v and u are illustrated in Figure 2.12.

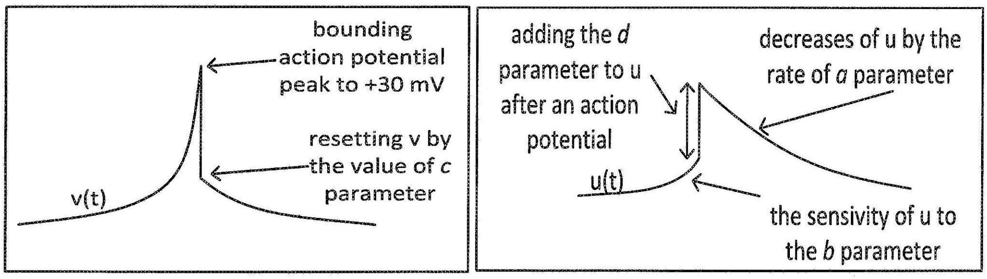

bounding action potential peak to +30 mV
resetting v by the value of c parameter
v(t)
adding the d parameter to u after an action potential
decreases of u by the rate of a parameter
u(t)
the sensivity of u to the b parameter

Figure 2.12 The effects of the parameters a, b, c and d on the dynamics of the variables v, u [10]

2.10 Artificial neural network generations

In general, artificial neural networks (ANNs) are categorized in three generation based on their computational units or the model of neurons that is employed in the neural networks. These generations are as the following:

2.10.1 The first generation of artificial neural networks

This generation of ANNs employs the model of neuron known as threshold logic unit which is the first artificial neuron model proposed by McCulloch and Pitts [18]. It contributed an important role in the developments of ANNs. For instances, multilayer perceptrons, Hopfield network and Boltzmann machines are the ANN architectures employing McCulloch-Pitts's neurons. This class of ANNs is capable of computing Boolean functions. In other words, the inputs and outputs are digital values either 0 or 1 and their processing units produce 1 if the sum of the inputs passes a threshold otherwise 0.The computation is modeled by an step function such as

$$f(x) = \begin{cases} 1 & \mathrm{x} \geq 0 \\ 0 & \mathrm{x} < 0 \end{cases} \tag{2.34}$$

2.10.2 The second generation of artificial neural networks

In this class of artificial neural networks, the processing units utilize activation functions that map their inputs, the weighted sum, to a defined rage of continuous values as their outputs. Linear saturated π (2.35) and Sigmoid σ (2.36) functions are the common activation functions. Typical networks of this generation are feed-forward neural network, recurrent neural network and radial basis functions. Moreover, the second generations of ANNs are able to compute any continuous function. It has been shown [34] that second generation neural networks can compute certain Boolean functions by using less neurons than those of the first generation and more importantly learning methods based on gradient descent can be used in this class of ANNs.

$$\pi(net_i) = \begin{cases} 1 & net_i > 1 \\ net_i & 0 \le net_i < 1 \\ 0 & net_i < 0 \end{cases} \quad \& \ net_i\text{= weighted sum of the inputs to neuron } i \tag{2.35}$$

$$\sigma(net_i) = \frac{1}{(1 + e^{-net_i})} \tag{2.36}$$

The second generation ANNs advanced the computational capability of artificial neural networks compared to the first generation. However, in terms of the biological observation the output the second generation neurons, for instance sigmoid neuron, are somehow close to the mean firing rate of a biological neurons and the mean firing rate is the least operational mode in biological neurons.

2.10.3 The third generation of artificial neural networks

This generation of artificial neural networks known as spiking neural network employs spiking neuron as their computational processing units. In the previous section we covered the two main models of this family of neurons and there are more than eleven spiking neuron models which are the variations of the two main models [16].

The inputs and outputs of this class of neural networks are in the form of spikes fired in certain time and the network process information based on the timing of each individual spike. Representing information in temporal coding and processing information based on the timing that neurons firing spikes are the most likely operation modes in real biological neural networks.

One of the simplest spiking neuron employed commonly by the third generation ANNs is the Spike Response Model with order zero (SRM_0). The state of a neuron described by its membrane potential $u(t)$ and the threshold level θ at the time t are

$$u_i(t) = \sum_{j=1}^{n} w_{ij}.\varepsilon(t - t_j^f - d_{ij}) \tag{2.37}$$

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau}.e^{1-(\frac{t}{\tau})} & t > 0 \\ 0 & t \le 0 \end{cases} \tag{2.38}$$

$$\text{if } u(t) \geq \theta(t - t_i^f) \text{ then } t_{i+1}^f = t \tag{2.39}$$

In the equation (2.37) pre-synaptic neurons (j=1,...,n) send spikes to neuron *i*. Function ε describes the post synaptic potential (PSP) response and the constant τ defines decay time of PSP. The synaptic delay between neuron *j* and neuron *i* denoted d_{ij}. Moreover, according to (2.37) a linear weighted sum is computed over all PSPs caused by the firing of the pre-synaptic neurons up to time *t* to determine the membrane potential of neuron *i* at time *t*. In addition, the last firing time t_i^f of neuron *i* is updated to the most recent time whenever neuron *i* fires. This is the time that membrane potential passes the threshold level from below which is equal to $\theta(t - t_i^f)$ as shown in (2.39).The dynamics of the threshold and PSP response function has been illustrated in the section 2.6.2.

2.11 Spike timing encoding methods

By spike timing encoding methods, the sensory data mostly analog are transformed to the timing of spikes, or spike train. Converting the data to the temporal form enables spiking neural networks to process numeric information represented to the networks. The two spike timing encoding methods are as the following [35]:

2.11.1 Sparse coding

In the context of biology, there is a sensory space known as receptive field where a stimulus is represented to the area depending which part is affected by the stimulus certain sensory neurons are excited. Therefore, to generate firing time the Gaussian RF is employed in sparse coding to simulate the procedure of converting continuous data to firing time.

In sparse coding the minimum and maximum of the input data is determined and a population of Gaussian RF neurons m chosen each for a certain range of the continuous input data. The center C of each Gaussian RF neuron and the width σ are calculated such as

$$c_i = \min + \frac{2i-3}{(m-2)(\max - \min)} \qquad i=1,\ldots,m \text{ (number of Gaussian RF neurons)} \tag{2.40}$$

$$\sigma = \frac{1}{\beta(m-2)(\max - \min)} \qquad \beta = \text{a constant value} \tag{2.41}$$

The method creates $n \times m$ real values between 0, 1 and each value triggers the corresponding neuron at time t .For example in a time interval of 10 ms a neuron is trigger at early time t=0 by the value from 0.9 to +1 and the other neuron at late time t=10 by the value from 0 to 0.1.If a spiking neural network has n nodes for n input, by sparse coding the input nodes are extended to $n \times m$. The firing times of two Gaussian RF neurons for the input value of 50, 100 in the time interval of 10 ms is shown in Figure 2.13.

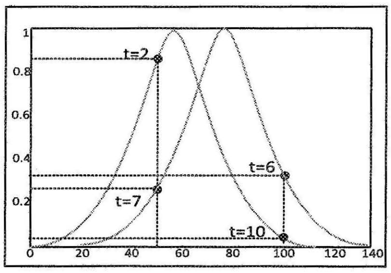


Figure 2.13 Converting the numeric values from 0 to 140 to firing time by two RF neurons in 10 ms interval [35]

2.11.2 1-D coding

In this type of spike firing encoding scheme, each input real value is mapped to a time interval [*a*, *b*] called "encoding interval" through a linear function (2.42).The advantage of the method is that the number of input neurons and consequently weights in a spiking neural network are not increased, unlike sparse coding method.

$$f(x) = \frac{(b-a)}{(\max - \min)} \times x + \frac{(a \times \max - b \times \min)}{(\max - \min)} \qquad (2.42)$$

The minimum and maximum are the boundaries of the input data which are converted to the firing times.

CHAPTER 3: METHODOLOGY

In this chapter, we describe the proposed spiking bidirectional associative memory (SPBAM). The structure of the proposed SPBAM is similar to that of the standard BAM which is a two-layer recurrent artificial neural network where each neuron in the first layer (*X*) is fully connected to the all neurons in the second layer(*Y*) and vice versa. In addition, each layer can act as an input or output layer to redirect the output of one layer to the other one.

The spiking neurons employed for the SPBAM are based on Izhikevich's cortical spiking neuron model [10]. According to this spiking neuron model two types of neurons namely excitatory and inhibitory are defined, and different classes of neurons can be also selected within each type. This variation is possible by setting certain values for the neuron's parameters which were discussed in Chapter 2.

There are twenty classes of spiking neurons [12] including Tonic Spiking, Phase Spiking, Tonic bursting and etc. The discussion of the classes in detail needs more solid background in biology and is beyond this study. However in brief, each class is known by the spiking pattern that the cortical spiking neurons are able to generate based on the presence of stimulus and the current state of neuron.

These twenty known spiking patterns or neuro-computational patterns are believed to contribute significant roles in temporal coding and spike-timing information processing. Therefore, based on these important neural features, the proposed SPBAM has a set of heterogeneous neurons that consists of different types and classes, and this set of neurons is selected by means of Genetic Algorithm optimization method.

3.1 Firing-time patterns

An important part of the SPBAM is the way that information is presented to the neural network. In standard BAM, the information of the patterns is presented in the form of n-dimensional vectors in the Hamming space. However, the form of representing patterns to the SPBAM is based on the timing that action potentials are scheduled for the input neurons. An action potential is a pulse that a spiking neuron emits when its membrane potential reaches a certain voltage threshold. Therefore a pattern in the form of an n-dimensional vector is converted to a set of spikes that are sent to input neurons in the pre-defined timings.

To clarify the process of converting each patterns to spike-timing form, we assume $\vec{A}$ is a k-dimensional vector in the Hamming space such as

$$\vec{A} = (a_1, a_2, \ldots, a_k) \quad \& \quad a_i=+1 \text{ or } a_i=-1 \text{ for } i=1,\ldots,k \tag{3.1}$$

At the beginning, a T-step time window is defined. That is for each component of the pattern a pulse or an action potential is scheduled in a certain time within the time window. Thus if at time t a spike has been scheduled for the i^{th} component, a_i, a pulse is sent to the neuron i corresponding to the i^{th} component.

For instance, in Figure 3.1 is shown that for a_1, a_2 and a_k corresponding to the neurons 1,2,k three spikes are generated at time 2,5 and 10, and at the time first neuron receives a spike, the second neuron and k^{th} neuron do not receive any spike.

One issue to convert the input data to spike-timing form is that how to schedule the timing of spikes. This depends on the type of information that is used as input for a spiking neural network. In general, there are two methods which were discussed in Chapter 2.

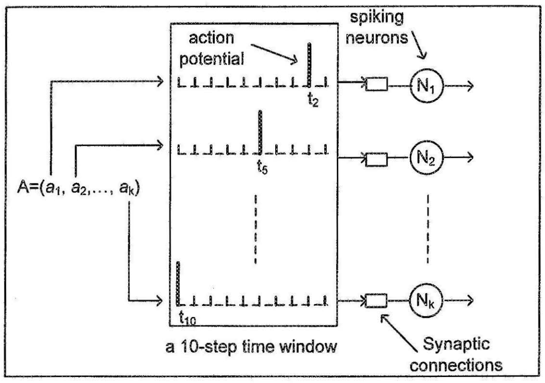


Figure 3.1 An example of representing a *k*-dimensional pattern *A* to spike-timing form in a 10-step time window

As mentioned in standard BAM, the information of the patterns is coded in *n*-dimensional vectors which their components are either +1 or -1 where *n* is equal to the number of neurons in each layer. Thus, we can schedule each +1 as an early firing and -1 a late firing. In other word, we divide a *T*-step time window to two parts, and for each +1 value an action potential is assigned at random time within the first half of the time window. Accordingly, for each -1 value an action potential is scheduled within the second half of the time window at random time. This process is repeated for all *n* components of each pattern. For instance, having a pair of patterns such as *A*=(-1,1,1,-1) ,*B*=(1,-1,-1,1), we can generate the spike-timing association as shown in Figure 3.2.

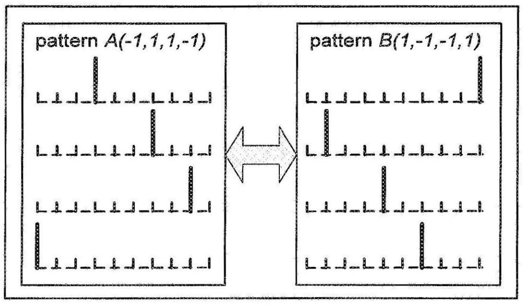


Figure 3.2 The association of two patterns *A,B* in spike-timing form

3.2 The spiking BAM training

To train the proposed SPBAM, we have applied the Genetic Algorithm optimization method, taking advantage of its outstanding exploration and exploitation properties, and the co-evolution method. The common training methods such as back-propagation and Hebbian learning are not adaptable for the training of the spiking BAM that employs Izhikevich's spiking neuron model since the training methods do not have any instruction and rule to choose the proper values for the neuron's parameters except the weights adjustment rules. Therefore by only weights adjustments the neural network is not trainable.

The learning starts with two populations that evolving separately at the same time, and then the individuals of the two populations are integrated to form a complete spiking BAM for evaluation processes. The two populations are the weight and the neuron population. Each individual's size in the weight population is equal to $m \times n$ where m denotes the dimension of the patterns in X layer and n is the dimension of patterns in Y layer. The values of weights are selected randomly from [-1, +1].

The neuron population is the second population that every individual in the population have totally $m+n$ neurons for both X and Y layers. Moreover, each neuron has six parameters. These parameters are a,b,c,d ,described in Chapter 2, and two added parameters as the following :

- The parameter of time step, η, in millisecond defining the changes of the variables u, v.
- The type T defining the neuron's type which is either excitatory or inhibitory.

The structures of each individual in the two populations are shown in Figure 3.3.

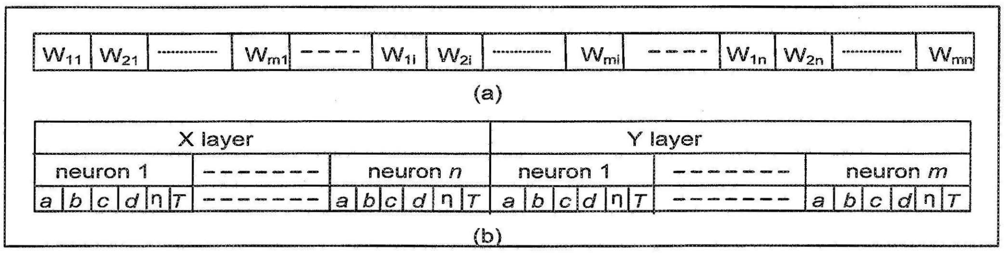


Figure 3.3 The structures of each individual in weight population (a) and neuron population (b)

The randomly selected values for the parameters *a,b,c,d* are within the ranges that have been defined in [10]. The value of the parameter η is chosen from [0, +3] with the interval of 0.05. The parameter T is defined based on the values of *a, b, c* and *d*. That means choosing the values in certain ranges defines that a neuron is inhibitory or excitatory.

3.3 The SPBAM fitness evaluation

During the evolutionary training two individuals from the weight and neuron populations are selected and integrated to form a possible solution that can be the desired spiking BAM if its fitness evaluation meets the minimum error. The network's fitness is based on the timing errors that occur between the actual output and the desired timing pattern. For example, let assume that we aim to associate the timing pattern T_a to the timing pattern T_b. After constructing the SPBAM from the selected individuals, T_a is presented to the network's Y layer and the output T_b' is obtained. Thus, one part of the network total error would be the timing differences between T_b and T_b' denoted by E_y. Then the timing pattern T_b is presented to the network's X layer and the output T_a' is obtained. Similarly, the rest of the network total error is calculated by the timing differences between T_a and T_a' denoted by E_x. The network total error for the two patterns T_a and T_b is equal to

$$E=E_x+E_y \tag{3.2}$$

The total timing errors for the spiking BAM is equal to

$$\sum_{k=1}^{P}\left(\sum_{i=1}^{n} Ex_{ik} + \sum_{j=1}^{m} Ey_{jk}\right) \qquad k=1,\ldots,P\,,\ i=1,\ldots,n\ ,\ \ j=1,\ldots,m \tag{3.3}$$

where P is the number of timing patterns and n , m are the number of neurons in X and Y layers respectively.

As shown in Figure 3.4, the aim is to train the spiking BAM having one action potential in the output of one neuron within the time window after the input timing pattern completely presented to the network. In case that only one spike occurs at the output neuron the error is equal to the difference between the desired and actual firing times. However, during the training we can expect no spike or more than one spike at the output neuron. In order to prevent such occurrences, during fitness evaluation such an individual is penalized by assigning a maximum error equal to the length of the time window.

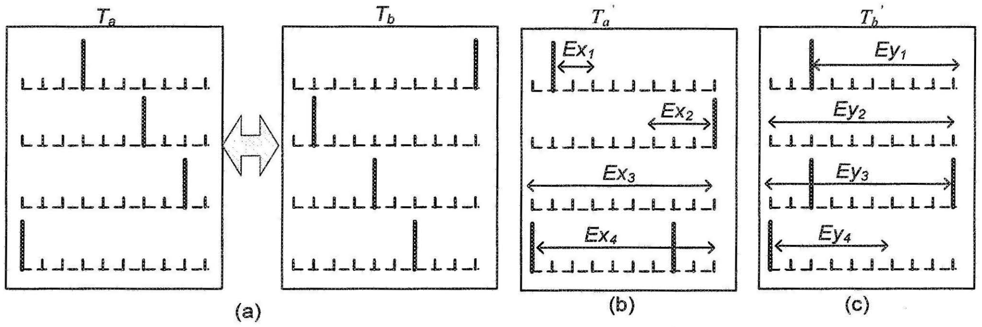


Figure 3.4 (a) Association of two timing patterns T_a, T_b.(b)Timing errors between T_a(desired output), T_a'(actual output), (c) Timing errors between T_b(desired output), T_b'(actual output)

3.4 The states of the spiking neurons

An important issue during the training is that how the state of the spiking neuron should be updated. In this spiking neuron model there are two parameters that describe the state of a neuron at each timing step. The neuron membrane potential is denoted by v and the membrane recovery variable denoted by u. At the learning stage when a timing pattern T_a is presented within a T-step time window to the network , the states of all m input neurons in Y layer are updated according to the equations (2.32) ,(2.33) in each step of the time window regardless of the input neurons receiving spikes or not. In addition the current values of the state parameters are kept to calculate the new values of v, u at the next step of the time window. The same process is followed when the timing pattern T_b is presented to the neurons of the layer X. During presenting a timing pattern to the network if a spike is scheduled to be sent at a specific step to a neuron, the amplitude of the spike +70 mV, which is a pre-synaptic pulse, is multiplied by synaptic weight. The result would be the amount of post-synaptic voltage that a neuron receiving from that pre synaptic neuron.

The training process explained here is used to associate only one pair of patterns. Thus, it is important to reset the state parameters to equilibrium states every time the next pair is presented to the network as follows

$$v = c \tag{3.4}$$

$$u = b.v \tag{3.5}$$

Otherwise, the order of patterns that are presented to the network would be depended to each other and this dependency is inappropriate and should be avoided.

3.5 Recalling the stored patterns

Ability to recall is another important property of BAM. Recalling the stored patterns basically is the process whereby we can retrieve one of the stored patterns together with its associated pattern even at the performance of noise. For instance, we assume that T_a, T_b are the two timing patterns stored and associated in the spiking BAM and T_{na}, T_{nb} are the noisy timing patterns of T_a and T_b respectively with certain noise. Thus, the aim is to obtain T_a, T_b from T_{na}, T_{nb} using the recalling mechanism. In standard BAM a pattern with certain noise means the Hamming distance between a pattern and its original one. However in spiking BAM a noisy version of a timing pattern means the firing-time differences between the pattern and its original timing pattern.

As in the standard BAM to recall a pattern , we need an error correction method. Let's assume that Tx_1 and Ty_1 are the noisy versions of the stored patterns shown in Figure 3.5(a) which are presented to the layers X, Y of the network. At the beginning, Tx_1 is given to the Y layer as an input and the output timing pattern Ty_2 is obtained from the output of layer Y.

After applying the error correction procedure on Ty_2, the result which is the modified timing pattern Ty_2 is given to the layer X as an input and Tx_2 is acquired from the output of the layer X. Similarly the error correction procedure is applied on Tx_2 for possible modification. One such a cycle that the information is sent back and forth between the two layers is called a reverberation or iteration. This cycle is continued each time with the new outputs of the two layers until there is not any change between the currently obtained outputs and those of the previous iteration for the layers X and Y.

In other word the spiking BAM reaches an stable point at the iteration i when there is not any firing-time difference between the two outputs Tx_i and Tx_{i-1} and similarly for those of Ty_i and Ty_{i-1} as shown in Figure 3.5(b).

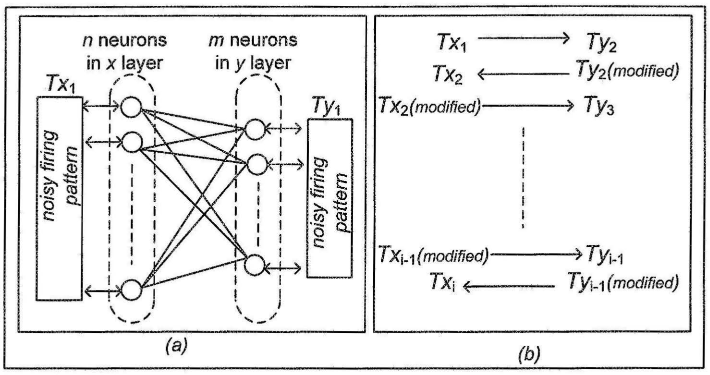


Figure 3.5 Representing noisy patterns to SPBAM (a), and recalling steps (b)

3.6 Error correction method

By presenting a noisy timing pattern, Tx_i , in a T-step time window at the iteration i to the spiking BAM, three situations may happen for the output pattern Ty_i , and accordingly the following error correction procedures are conducted to modify pattern Ty_i.

1- Output neuron k does not fire any spike: In this case, the firing time of k^{th} output (k=1,...,m where m= number of neurons in layer Y) from pattern Ty_{i-1} is scheduled for the k^{th} output of pattern Ty_i.

2- Output neuron k fires more than one spike: We expect a neuron fires only on spike within the time window. Thus, in this case we must choose only one firing time and eliminate the extra firings. To choose only one firing time for the k^{th} output of pattern Ty_i the distances of these firing times are measured from the k^{th} firing time of pattern Ty_{i-1}.Among all firing times the one is chosen that has the minimum distance from pattern Ty_{i-1}, and the rest of the firing times are removed.

3- Output neuron k fires exactly one spike: The firing time is assigned directly to the k^{th} firing time of pattern Ty_i, and there would not be any timing change to this firing time.

CHAPTER 4: EXPERIMENTAL SETUP AND RESULTS

In this chapter, we conduct four types of experiments and analyze the performance and capability of the proposed spiking bidirectional associative memory (SPBAM) in several aspects. In addition to providing a sound justification, we have compared our results to those of the standard BAM with the similar experimental settings.

The outlines of the experiments are listed as the following:

- SPBAM trainings.
- The SPBAM capability to recall.
- The SPBAM convergences.
- The speed of SPBAM convergence.

In all experiments, the patterns used for the standard BAM are *n*-dimensional vectors in the Hamming space, and the patterns used for the SPBAM are firing-time patterns that are generated from the patterns that their specifications presented in the Table 4.1.

Table 4.1 The specifications of the training pattern sets

Training pattern set	Number of patterns	Pattern X's dimension	Pattern Y's dimension
A	4 pairs	12×12	5×11
B	6 pairs	7×5	7×5
C	12 pairs	7×5	7×5

The graphical forms of the training pattern sets *A*,*B* and *C* are provided in Appendix A. The above training pattern sets have been converted to the firing time patterns which are in six-step and twelve-step time window. Therefore, the experiments related to the spiking BAM are repeated twice since the timing patterns generated in two different lengths of time window. The reason is to explore the performance of the SPBAM when the firing-time pattern represented in different lengths. An example of firing-time pattern in twelve-step timing length which belongs to the first pattern pair of the training pattern set *C*, is given in Appendix B.

4.1 The spiking BAM trainings

The spiking BAM is trained with six different firing-time patterns generated from the training set *A*, *B* and *C* in six and twelve timing steps. The trainings results are shown in Table 4.2.

Table 4.2 Training results of the spiking BAM with the firing-time pattern sets

Training pattern set	Maximum expected timing error in 6,12-step time window	Minimum timing error in 6 ,12-step time window after training	Training accuracy
A	4776 , 9552	141 , 328	%97.04 , %96.56
B	2520 , 5040	67 , 156	%97.34 , %96.90
C	5040 , 10080	398 , 907	%92.10 , %91.00

The maximum timing error for the training pattern set *P* is equal to

$$\text{Number of pattern pairs} \times \text{Length of time window} \times (X\ \text{dimension} + Y\ \text{dimension}) \qquad (4.1)$$

In all trainings, the results have been recorded when the trainings were performed from 3.10^4 to 5.10^4 generations in 10 runs. Figure 4.1 and 4.2 show the trainings of spiking BAM with the firing-time pattern sets *B* and C in six-step and twelve-step time window respectively. The other four training graphs are provided in the Appendix C.

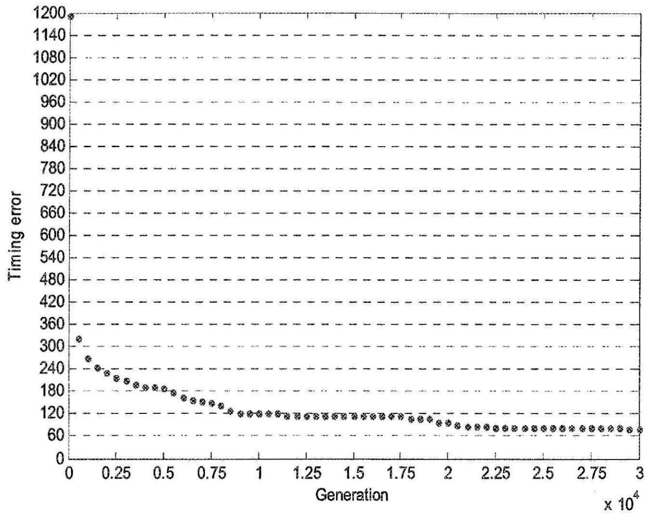


Figure 4.1 Training of pattern set B with six-step time window

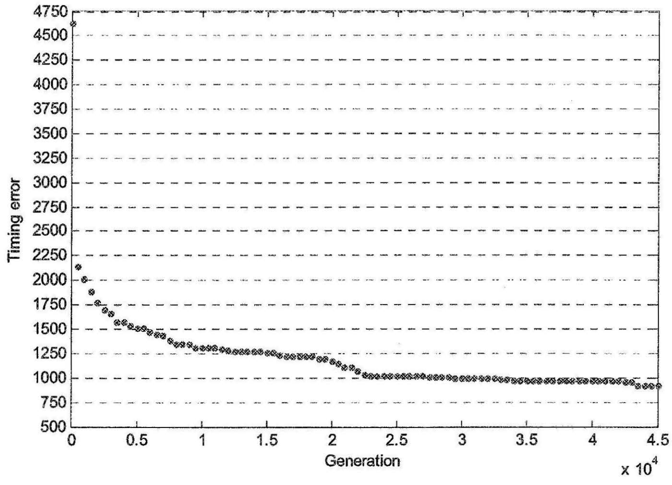


Figure 4.2 Training of pattern set C with twelve-step time window

4.2 The spiking BAM capability to recall

In this section we conduct six experiments. In each experiment the trained SPBAM is tested against the corresponding firing-time pattern. The results are compared to those of the standard BAM. There are three firing time patterns that each one represented in two different lengths of time window. The level of noise applied to the timing patterns ranges from 0-25%.

Noises are applied on both patterns which are associated to each other in layers *X*, *Y*. To apply noise on a pattern, a row in the firing time pattern is selected. If the original firing-time scheduled at the first half of time-window we change this firing time to a random time within the second half of the time-window and vice versa. The procedure of adding noise is repeated for the other rows based on the noise percentage.

In the next step, all patterns of the training pattern set are presented to the network and the error correction process is applied on each pattern as discussed in the Chapter 3 until the network reaches a stable point. The final firing time patterns, for each pair, are converted to *n*-dimensional vectors consisting of +1 and -1 to count the number of mismatches with their original patterns.

To evaluate the recall procedure for both the SPBAM and standard BAM, the results are compared for the both networks as shown in Figures 4.3-4.8. The experimental results belong to the firing time patterns in six-step and twelve-step time window.

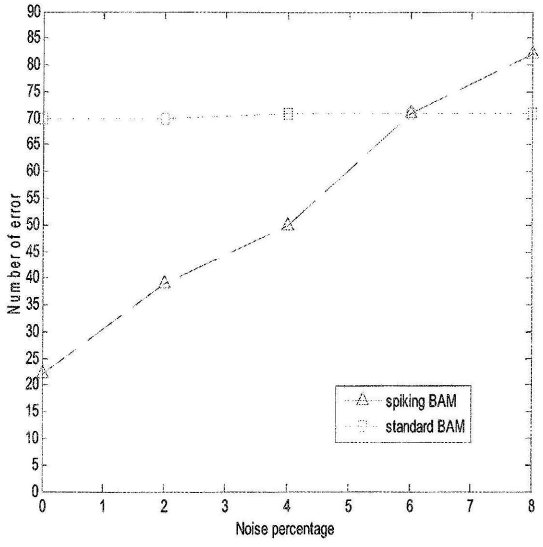


Figure 4.3 Recalling pattern set *A* in 6-step timing

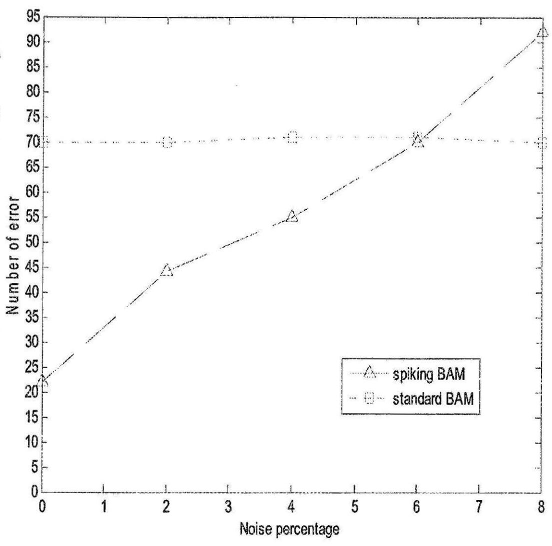


Figure 4.4 Recalling pattern set *A* in 12-step time timing

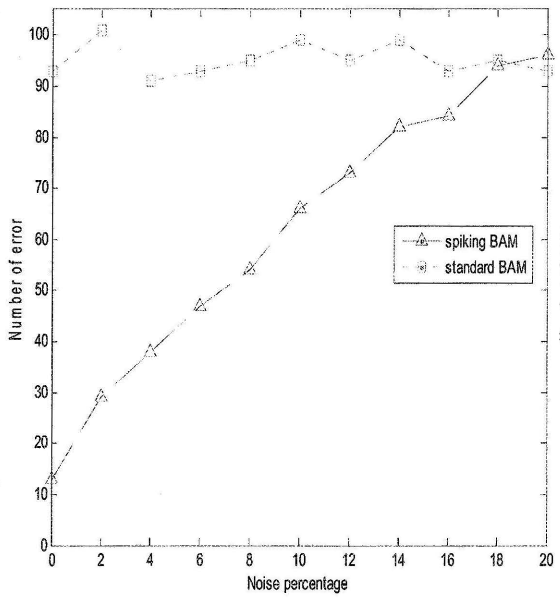


Figure 4.5 Recalling pattern set *B* in 6-step timing

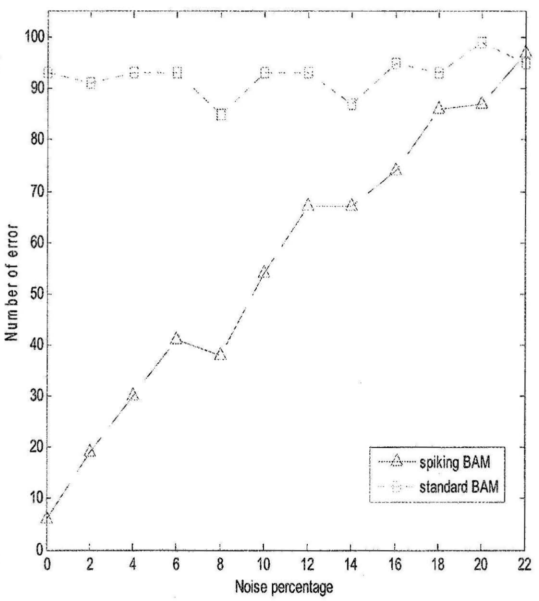


Figure 4.6 Recalling pattern set *B* in 12-step timing

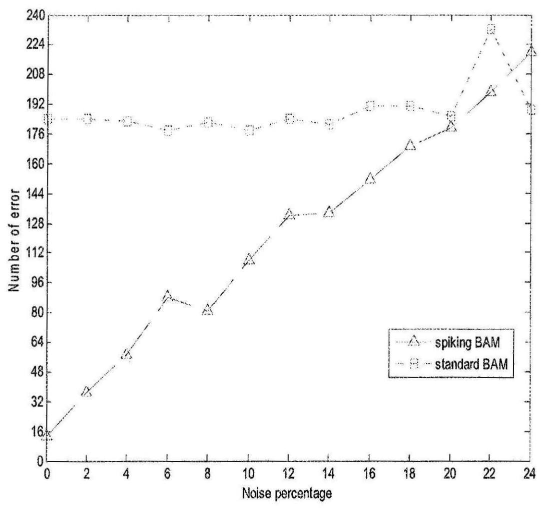


Figure 4.7 Recalling pattern set *C* in 6-step timing

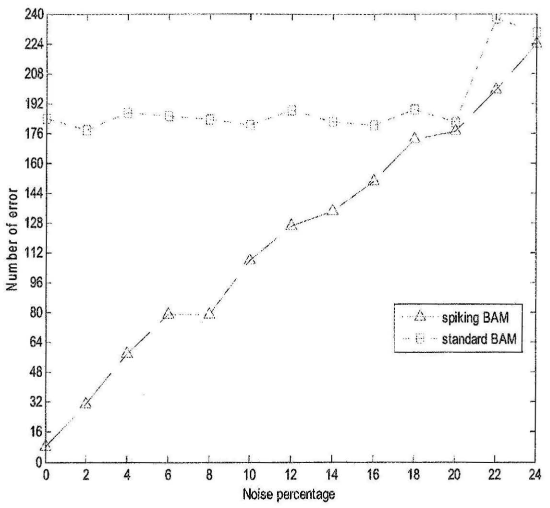


Figure 4.8 Recalling pattern set *C* in 12-step timing

4.3 The SPBAM convergences

In this section we have provided the results that show the convergence of SPBAM on all three firing time pattern sets in two different time windows. To describe the experiments, a pattern pair is selected randomly from each pattern set. A random noise from 5% to 30% is added to the both patterns in layers *X*, *Y*. These noisy timing patterns are presented to the SPBAM. The error correction process is applied until the network reaches a stable point.

During the recall for each reverberation, the timing differences between the current and previous outputs are measured in both layers. After the network reaches a stable point when there are no further changes in the outputs, we continued the recall for extra ten iterations to justify the network convergence for any randomly selected patterns. The network convergences on randomly selected timing patterns are shown in Figures 4.9, 4.10.

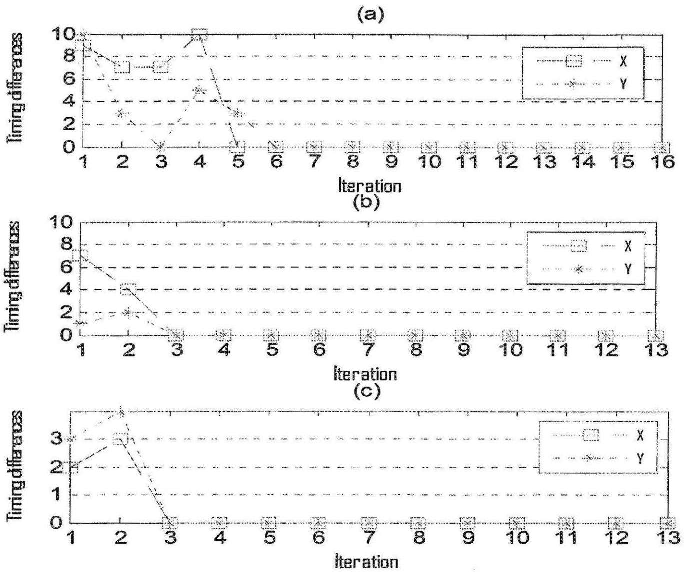


Figure 4.9 The spiking BAM convergences on pattern sets A, B, C with six-step time window

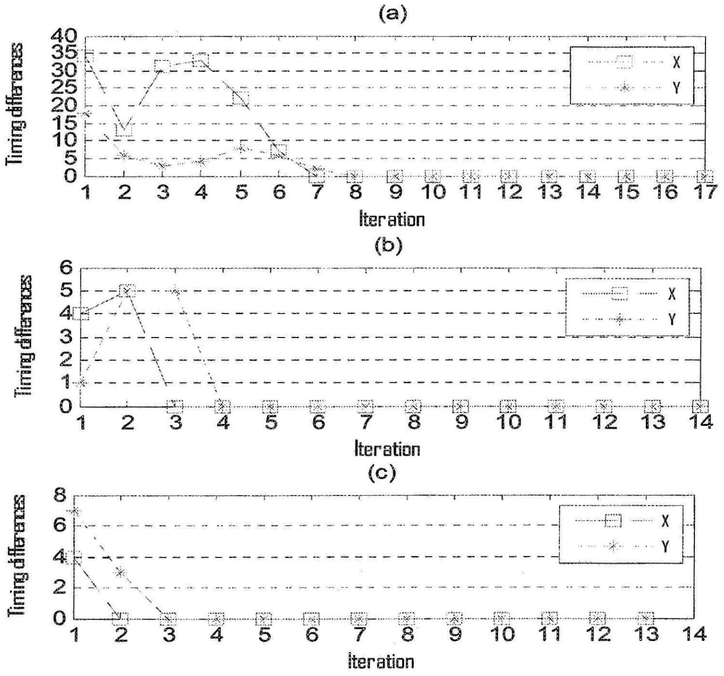


Figure 4.10 The spiking BAM convergences on pattern sets A, B, C with twelve-step time window

4.4 The speed of SPBAM convergence

In this section the two networks, spiking BAM and standard BAM, are explored for the number of iterations during recalls. Similar to the previous experiments, we examine the issue on all three pattern sets which are presented in two different time windows. For each experiment, all patterns belonging to one pattern set are presented to both networks and the total iterations to recall each pattern set are counted until the networks being stabled. Evaluating the number of iterations is repeated for different range of noises. Figures 4.11-4.13 demonstrate the results of these experiments.

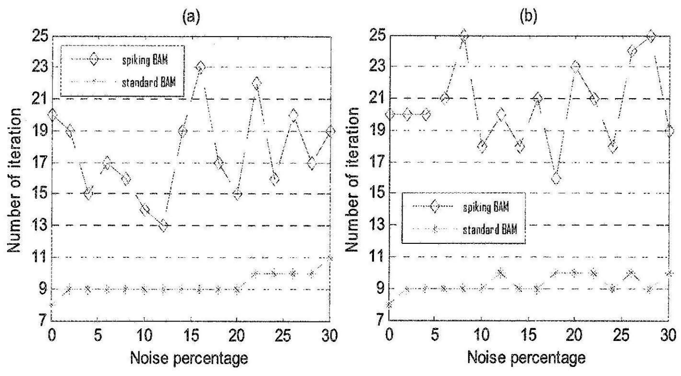


Figure 4.11 Total iterations on the pattern set *A* represented with six (a) and twelve (b) steps timing

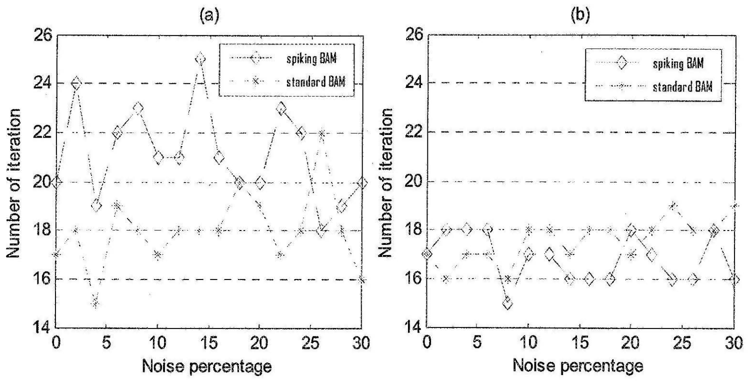


Figure 4.12 Total iterations on the pattern set *B* represented with six (a) and twelve (b) steps timing

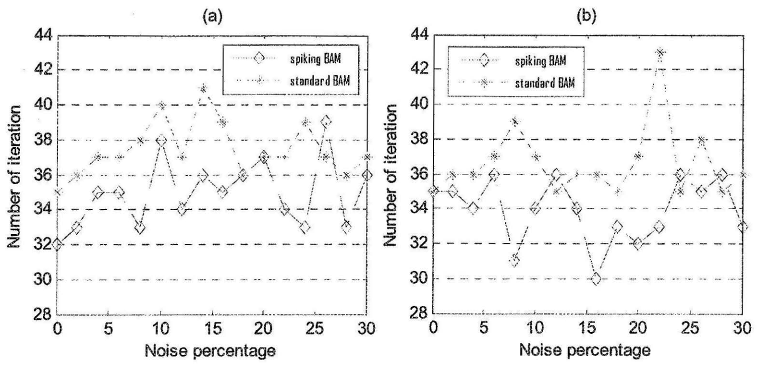


Figure 4.13 Total iterations on the pattern set *C* represented with six (a) and twelve (b) steps timing

4.5 Recalling patterns with noise in one layer

In this section, we study the capability of recalling patterns by the SPBAM when the noises are applied only in one layer either X or Y .The same experiments are also conducted with standard BAM and the results are compared. The description of the experiments is similar to those explained in the section 4.2 except the noise is added randomly to one layer. Figures 4.14-4.16 show the outcome of the experiments.

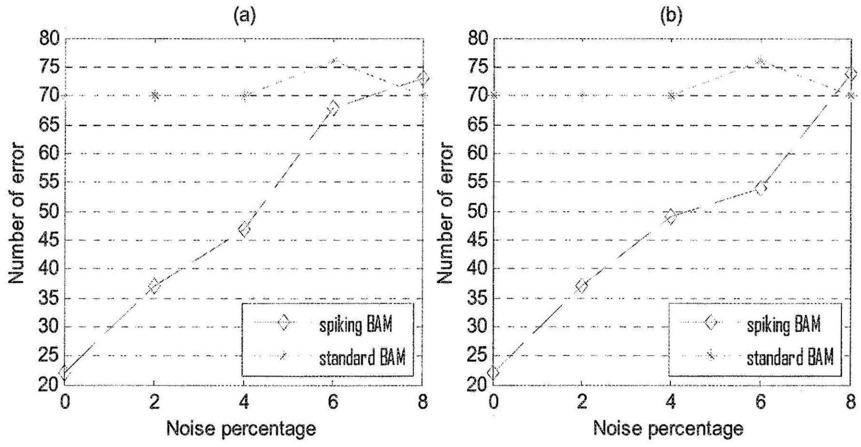


Figure 4.14 Recalling pattern set *A* in 6-step (a) and 12-step (b) time window with noise added on one layer

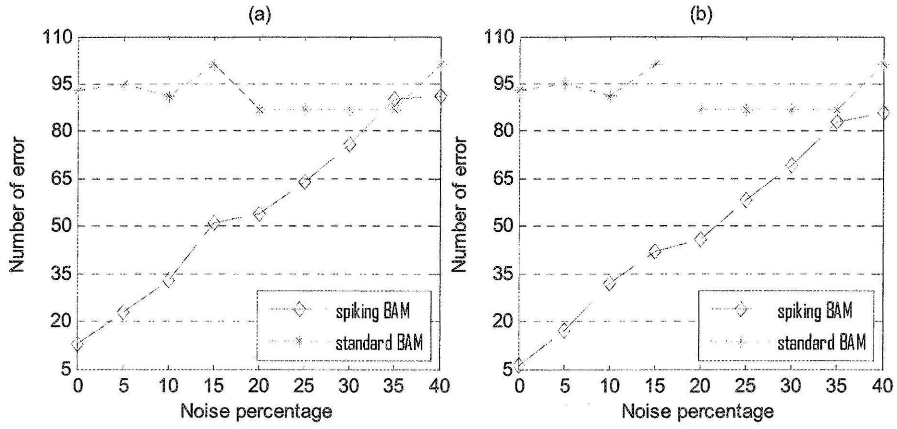


Figure 4.15 Recalling pattern set *B* in 6-step (a) and 12-step (b) time window with noise added on one layer

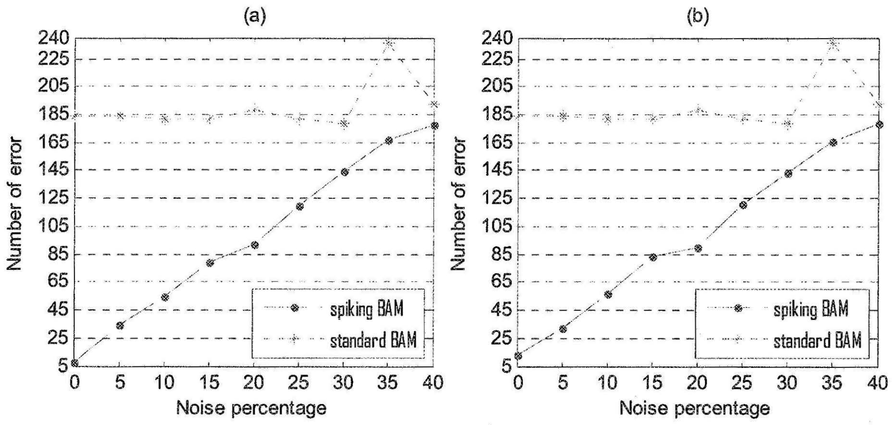


Figure 4.16 Recalling pattern set *C* in 6-step (a) and 12-step (b) time window with noise added on one layer

4.6 Discussion

Using to the experimental setups, we aimed to investigate several main features of bidirectional associative memory for the proposed spiking BAM. To evaluate the performance and capability of the spiking BAM, the experimental results were compared to those of the standard BAM. The features of interest are training, storage capacity, recall and convergence.

The spiking BAM's training is based on GA and Co-evolution methods. This type of evolutionary optimization is computationally expensive. However, the spiking neuron model employed in the spiking BAM leads to select such a method for the trainings since training methods such as Hebbian learning does not have rules for adjusting the neuron's parameters. Therefore, by adjusting only weights the network can't be trained. On the other hand in standard BAM the weights are calculated based on the prescription method discussed in the Chapter 3.The training results obtained for SPBAM show that although increasing the pattern dimensionalities would increase the computational cost, however this does not hamper the training accuracy. As shown in Tabel 4.1, the training process affected mainly by the number of patterns in each training pattern set and to a lesser extent by the length of the time-window defined for the training firing-time patterns.

The storage capacity of the spiking BAM is much higher than the standard BAM. By comparing the two networks results depicted in Figures 4.3-4.8, it can be observed when the patterns are represented without noise, the stored patterns are recalled more accurately by spiking BAM than the standard BAM. It is also noted that this difference increases significantly when the number of patterns increase such as in pattern set *C*.

To investigate the quality of the spiking BAM's recalls, we provide results that confirm spiking BAM surpasses the standard BAM especially when the number of patterns increases. In addition to out-performing the standard BAM when higher number of patterns exists, the spiking BAM also continues to recall the patterns more accurately in the presence of higher percentage of noise. This can be confirmed using the results shown in Figures 4.3-4.8. Moreover, when the noise is only applied to the patterns of one layer, the recall performance increases significantly (up to forty percent of noise) as shown in Figure 4.14-4.16. Meanwhile, the experiments show either with one or two layers noise, the network recalls more accurately when the timing patterns presented in time window of double length.

Another issue in the bidirectional associative memory is the network's convergence. In the experiments of section 4.3, we provided results that show the proposed spiking BAM is able to converge to a stable point when the recall process takes place. In the experiments shown in Figure 4.9-4.10, after the networks reaches a fix point, we continued ten more iterations for each test to verify that there is not any further changes for the state of the spiking BAM.

The last feature that we studied in section 4.4 is the number of iterations that either spiking BAM or standard BAM needed during recall when all patterns for pattern set *A*, *B* and *C* are tested. Figures 4.11 shows that when the dimension of the patterns are significantly higher than the number of patterns such as pattern set *A* , the standard BAM converges with less number of iterations and this trend continues even the firing time pattern presented with 12-step time window. However, when the difference between the pattern dimensionality and the number of patterns decreases, the number of iterations that spiking BAM needs to converge decreases, and even with pattern set *C* we observe that the SPBAM outperforms the standard BAM. In addition the results improve more by choosing 12-step time window.

CHAPTER 5: SUMMARY AND CONCLUSION

5.1 Summary

Bidirectional associative memory (BAM) proposed by Kosko is an extended version of Hopfield associative memory that models the hetero-associative memory. The main application of the BAM is in pattern recognition. The BAM structure is a fully connected recurrent neural network that is able to associate, store and recall patterns. The current generations of Artificial Neural Networks although have been successfully applied for numerous engineering and scientific problems. However, the mode that current ANNs are processing information known as "mean-firing rate" or Rate coding is insufficient compare to the tremendously fast information processing of the cortex in the brain and it is contrary to the fact that the operational speed in the cortex is about 100 Hz.

Therefore, to improve the current artificial neural networks it is necessary to build the ANNs that their processing units (neurons) function as the biological neuron do. In the biological neural systems, neurons communicate with each other and the outside world by the means of action potentials or spikes and the timing of these spikes play the key roles in carrying and processing information.

The coding of information by action potentials is categorized in two main areas:

- Rate code, which aims to average the timing information over a time window of about 500 milliseconds.
- Spike code, which focuses on a timing of each single spike over relatively a short period of time about 10 milliseconds comparing to the Rate code.

The two coding schemes have several variations and the best coding scheme is the Temporal coding that carries the most information based on information theory as it was explained in Chapter 2. It is important to mention the Rate code is the extremely rare coding scheme that the biological neurons operate in the higher areas of the cortex. As a result, the third generation of ANNs known as spiking neural networks (SNNs) is a way to improve the performance of ANNs in general. To construct a spiking neural network, we need to employ spiking neurons. In this regards, several spiking neuron models have been proposed. Among the models, Hudgekin-Huxley and SRM are considered as the main categories. To construct a realistic model of SNN, two elements of a spiking neuron model affect the system

1. The computational cost of the spiking neuron model.
2. The ability of generating rich neuro-computational patterns (spiking patterns).

Spiking neural networks have been used for modeling Hopfield networks and feed-forward ANNs, and it has been found the later converging faster and requiring less neurons comparing to the second generation feed-forward neural networks. However the computational units employed in the networks are the simplified SRM neurons which have the lack of generating spiking patterns. To model a spiking neural network that fulfills the two mentioned criteria we have modeled a spiking BAM which employs Izeickevich's spiking neuron model. The information (patterns) is presented to the spiking BAM based on the temporal coding. The performances of the model for its training, recall, storage capacity, and convergence are investigated and the results of the experiments are compared to the standard BAM's.

5.2 Conclusion

The development of the third generation of artificial neural networks is in the initial stages. Although there are many issues and trades off in spiking neural networks(SNNs) regarding to their neuron models, coding schemes, structures, and training methods, but the evidences confirm that still SNNs are powerful than the ANNs of the second generation. More importantly their computational mode, based on our understanding up to now, is similar to biological neural networks. Therefore regarding the advantages of SNN, we proposed the spiking BAM and our experimental results confirm the better performances and capabilities in the number of areas that are interested for designing a bidirectional associative memory.

Nonetheless, the proposed model need more work on the subjects such as its training and its structures. Although the GA is a good solution for trainings, it suffers from the computational costs. It can be considered to adopt a training method for the spiking BAM. In addition, since we aimed to have a reasonable comparison between the spiking BAM and the standard BAM, we selected the same structure of the standard BAM, and it may be possible to investigate the spiking BAM with a different structure. A successful model of a spiking BAM would be a great candidate for engineering application such as neural prosthesis, and an interface between small population of neurons and electronic devices.

5.3 Future work

In order to achieve a reliable and optimum design for the proposed spiking BAM, we are interested to investigate and improve the model in the number of areas such as:

- Exploring a training method which is less computationally expensive than the evolutionary training method applied for the spiking BAM.

- Investigating a different architecture for the spiking BAM to reduce the number of links and neurons. This would improve the training time when the pattern dimensionalities increase.
- Modifying and evaluating the spiking BAM for colored patterns.

APPENDIX A: TRAININGS PATTERN SETS

A1 THREE PATTERN SETS

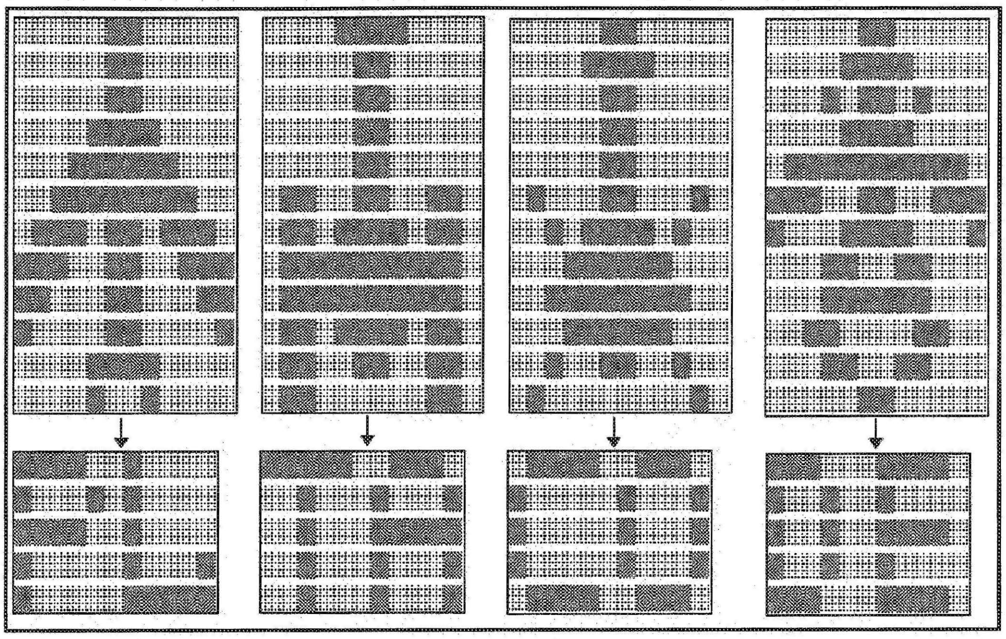


Figure A1 Pattern set *A* (4 pairs)

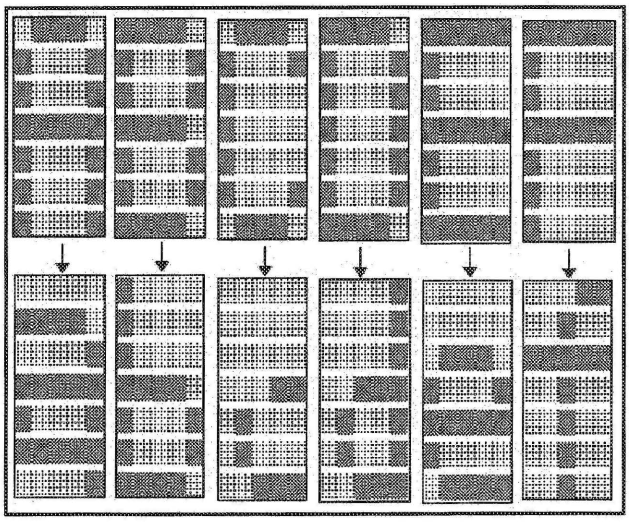


Figure A2 Pattern set *B* (6 pairs)

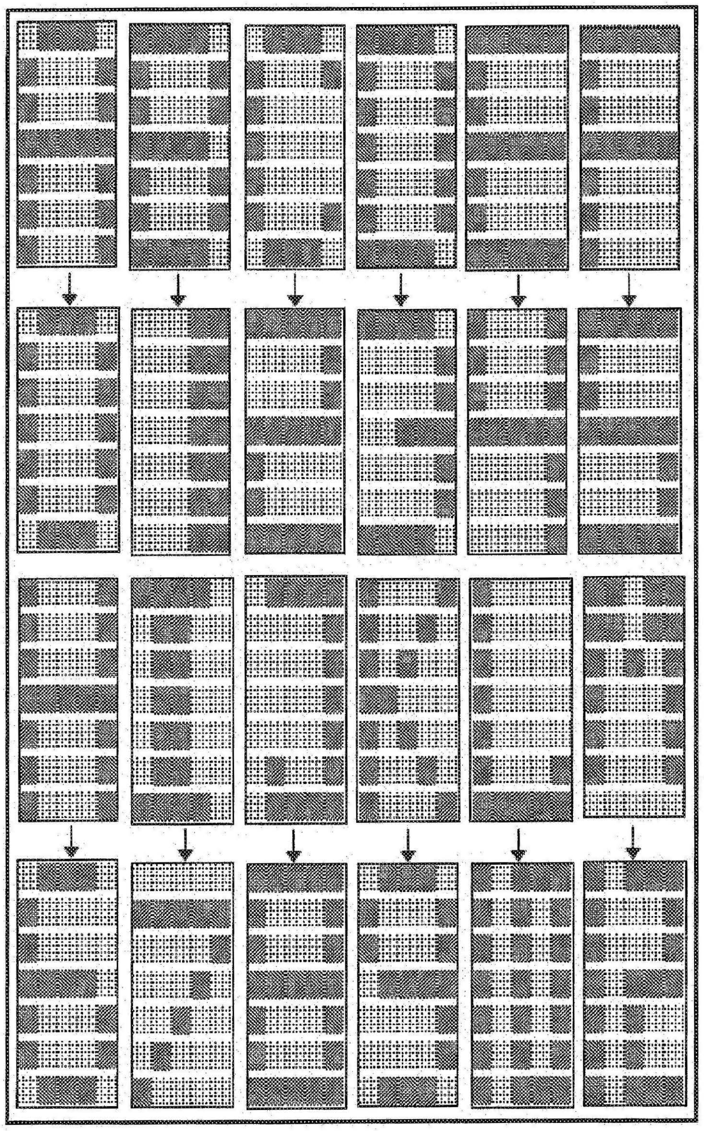
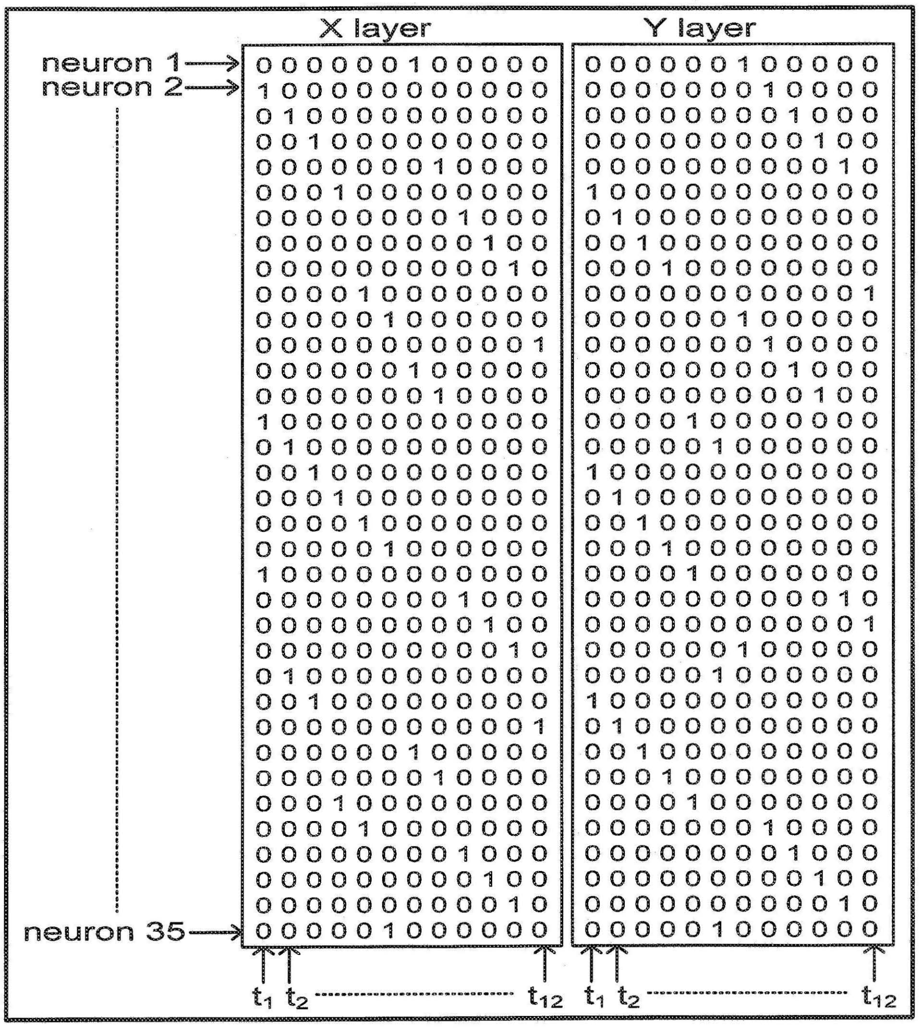


Figure A3 Pattern set *C* (12 pairs)

APPENDIX B: THE SAMPLES OF DATA

B1 FIRING-TIME PATTERN


neuron 1
neuron 2
neuron 35
t_1 t_2 t_{12}
t_1 t_2 t_{12}

X layer	Y layer
0 0 0 0 0 0 1 0 0 0 0 0	0 0 0 0 0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0	0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0	0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0	0 0 0 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1	0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0	0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0 0 0	0 0 0 0 0 1 0 0 0 0 0 0

Figure B1 Firing-time patterns with 12-step time window for the first pattern pair of the training pattern C

B2 Spiking neurons' parameters

Table B2 Spiking BAM neurons' parameters after training for the pattern set C (T stands for time step in millisecond, N stands for neuron's type, 0= Excitatory, 1=Inhibitory)

The values of neurons' parameters in X layer						**The values of neurons' parameters in Y layer**					
A	**B**	**C**	**D**	**T**	**N**	**A**	**B**	**C**	**D**	**T**	**N**
0.02	0.2	-63.47	7.39	0.9	0	0.02	0.2	-52.44	2.98	0.2	0
0.05	0.23	-65	2	0.45	1	0.04	0.24	-65	2	0.45	1
0.08	0.21	-65	2	0.35	1	0.02	0.2	-61.87	6.75	0.7	0
0.02	0.2	-64.32	7.73	0.35	0	0.02	0.25	-65	2	0.4	1
0.02	0.2	-56.69	4.68	0.45	0	0.05	0.23	-65	2	1.3	1
0.02	0.2	-62.22	6.89	0.3	0	0.02	0.2	-64.65	7.86	0.4	0
0.03	0.24	-65	2	0.15	1	0.02	0.2	-52.52	3.01	0.3	0
0.02	0.2	-64.82	7.93	0.45	0	0.07	0.22	-65	2	0.35	1
0.02	0.2	-53.47	3.39	0.4	0	0.02	0.2	-58.74	5.5	0.8	0
0.1	0.2	-65	2	1.05	1	0.02	0.2	-60.34	6.14	0.4	0
0.08	0.21	-65	2	0.35	1	0.05	0.23	-65	2	0.4	1
0.02	0.2	-64.85	7.94	0.3	0	0.02	0.2	-50.23	2.09	0.4	0
0.02	0.2	-65	8	0.55	0	0.02	0.25	-65	2	0.45	1
0.02	0.2	-61.24	6.5	0.85	0	0.1	0.2	-65	2	0.45	1
0.02	0.2	-61.79	6.71	0.65	0	0.07	0.22	-65	2	1.85	1
0.02	0.2	-64.9	7.96	1	0	0.05	0.23	-65	2	0.4	1
0.08	0.21	-65	2	0.85	1	0.02	0.25	-65	2	0.4	1
0.04	0.24	-65	2	0.25	1	0.05	0.23	-65	2	0.4	1
0.02	0.2	-65	8	0.55	0	0.04	0.24	-65	2	1.7	1

0.02	0.2	-63.9	7.56	0.8	0	0.02	0.2	-63.65	7.46	0.45	0
0.03	0.25	-65	2	0.3	1	0.02	0.2	-62.41	6.96	0.7	0
0.09	0.21	-65	2	0.25	1	0.06	0.22	-65	2	0.4	1
0.02	0.2	-62.29	6.92	0.3	0	0.03	0.25	-65	2	0.4	1
0.05	0.23	-65	2	0.65	1	0.06	0.23	-65	2	0.5	1
0.02	0.2	-59.1	5.64	0.75	0	0.02	0.2	-62.99	7.2	0.3	0
0.02	0.2	-64.22	7.69	0.45	0	0.03	0.24	-65	2	0.65	1
0.05	0.23	-65	2	0.45	1	0.06	0.23	-65	2	0.2	1
0.07	0.22	-65	2	0.4	1	0.03	0.25	-65	2	0.2	1
0.02	0.2	-63.87	7.55	0.65	0	0.09	0.2	-65	2	0.65	1
0.04	0.24	-65	2	0.35	1	0.06	0.22	-65	2	0.3	1
0.04	0.24	-65	2	0.35	1	0.02	0.2	-63.77	7.51	0.8	0
0.06	0.22	-65	2	0.9	1	0.02	0.2	-59.24	5.7	0.55	0
0.02	0.2	-55.51	4.21	1.25	0	0.02	0.2	-64.99	8	0.45	0
0.02	0.2	-63.43	7.37	0.55	0	0.02	0.2	-54.29	3.72	0.55	0
0.05	0.23	-65	2	0.3	1	0.03	0.24	-65	2	0.6	1

APPENDIX C: TRAINING DIAGRAMS

C1 FOUR TRANING DIAGRAMS

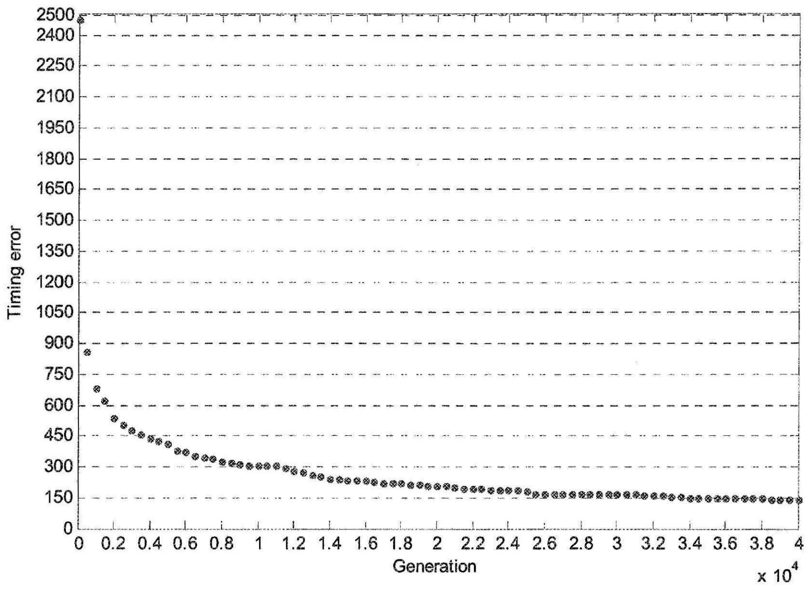


Figure C1 Training of pattern A with six-step time window

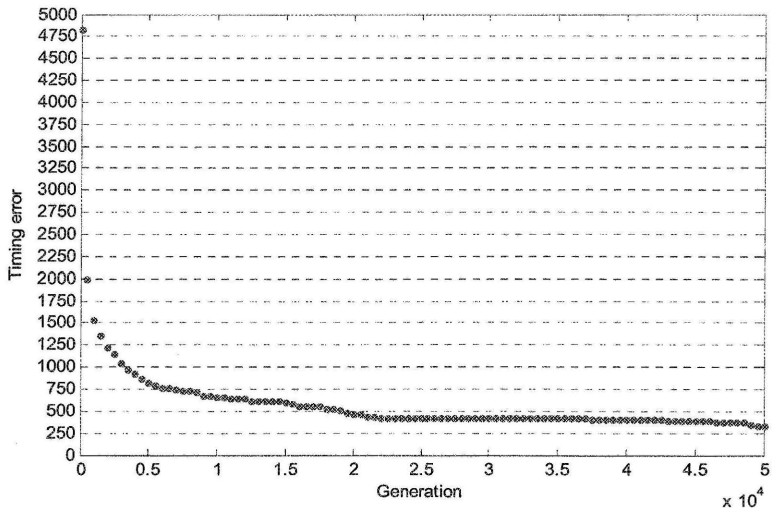


Figure C2 Training of pattern A with twelve-step time window

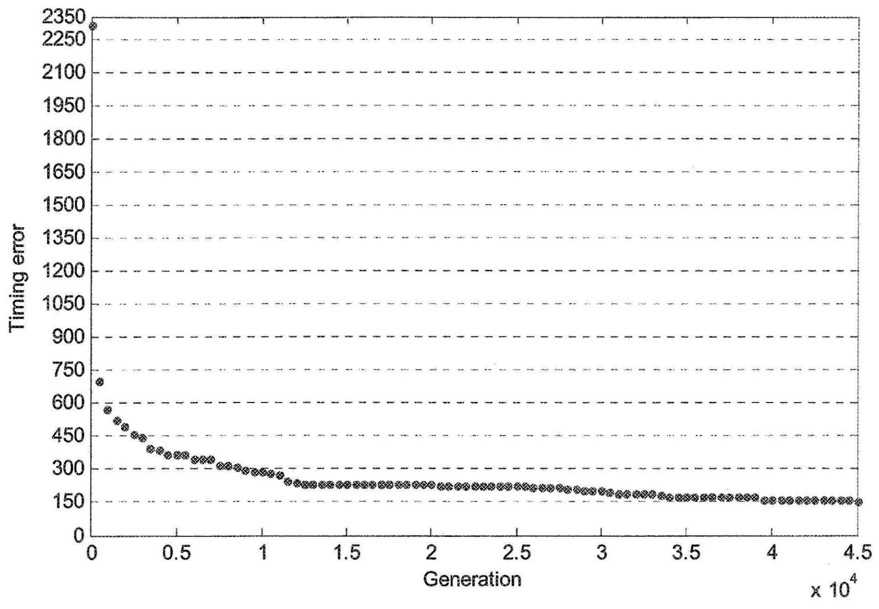


Figure C3 Training of pattern B with twelve-step time window

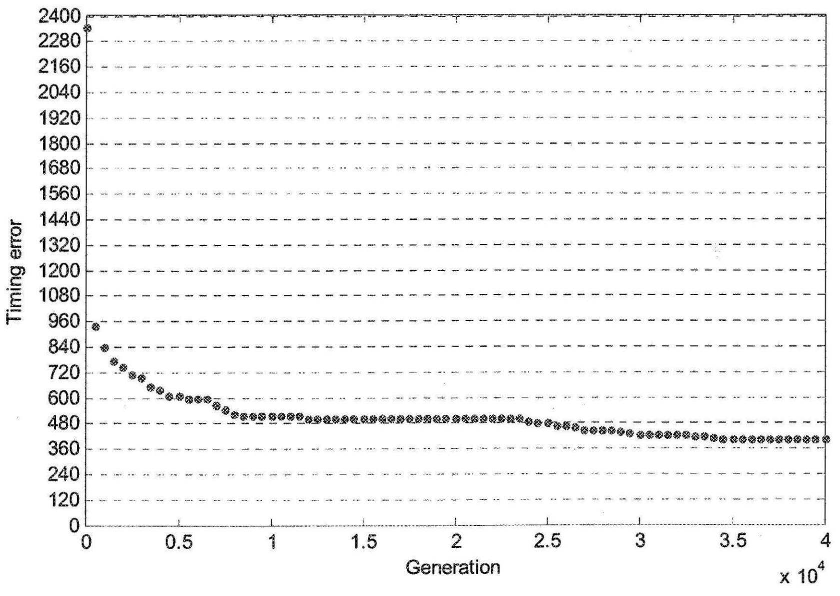


Figure C4 Training of pattern C with six-step time window

GLOSSARY

Action potential (spike) an electrochemical pulse generated by the biological neuron when its membrane voltage passes from certain threshold.

Associative memory a type of memory that is able to store patterns and retrieve them when the noisy forms of the stored patterns are presented to the memory.

Computational neuroscience the study of the brain functions which is related to the information processing in the neural system.

Firing-time pattern a pattern that is represented by the timing of action potentials in a time window.

Mean-firing rate a neural coding scheme that deals with a sequence of spikes generated in a period of about 500 milliseconds.

Neural coding a subject in neuroscience that investigates how sensory data or outside-world information is presented to the brain by neurons.

Neuro-computational patterns the type of firing-time patterns that the biological neurons generate when a stimulus is presented to the neuron.

Temporal coding a type of neural coding that is based on the exact timing of action potentials.

Time window a period of time that a set of information is presented to a neural network.

REFERENCES

[1] F. Rosenblatt, "Principles of neurodynamics: Perceptrons and the theory of brain mechanisms", *Spartan Books*, Washington, DC, 1962.

[2] J.H. Holland, "Genetic algorithms: Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand", *Scientific American*, vol. 267, pp. 66-72, 1992.

[3] E.D. Adrian, "The basis of sensation", *British Medical Journal*, vol. 1, pp. 287-290, 1954.

[4] F. Rieke, "Spikes: Exploring the Neural Code", *The MIT Press*, 1997.

[5] S. Nirenberg, and P.E. Latham, "Decoding neuronal spike trains: How important are correlations?", *Proc. of the National Academy of Sciences*, vol. 100, pp. 7348-7353, 2003.

[6] S.P. Strong, R. Koberle, R.R. de Ruyter van Steveninck, and W. Bialek, "Entropy and information in neural spike trains", *Physical Review Letters*, vol. 80, pp. 197-200, 1998.

[7] A.L. Hodgkin, and A.F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve", *Bulletin of Mathematical Biology, Springer*, vol. 52, pp. 25-71, 1990.

[8] H.R. Wilson, "Simplified Dynamics of Human and Mammalian Neocortical Neurons", *Journal of Theoretical Biology*, vol. 200, pp. 375-388, 1999.

[9] E.M. Izhikevich, "Resonate-and-fire neurons", *Neural Networks*, vol. 14, pp. 883-894, 2001.

[10] E.M. Izhikevich, "Simple model of spiking neurons", *IEEE Trans. Neural Networks*, vol. 14, no. 6, pp. 1569-1572, 2003.

[11] W. Gerstner, and W.M. Kistler, "Spiking Neuron Models: Single Neurons, Populations, Plasticity", *Cambridge University Press*, 2002.

[12] E.M. Izhikevich, "Which model to use for cortical spiking neurons?", *IEEE Trans. Neural Networks*, vol. 15, pp. 1063-1070, 2004.

[13] B. DasGupta, and G. Schnitger, "The power of approximating: a comparison of activation functions", *in Advances in Neural Information Processing Systems 5*, vol. 5, pp. 363-374, 1992.

[14] M. De Kamps, and F. van der Velde, "From artificial neural networks to spiking neuron populations and back again", *Neural Networks*, vol. 14, pp. 941-953, 2001.

[15] E.T. Rolls, and M.J. Tovee, "Processing speed in the cerebral cortex and the neurophysiology of visual masking", *Proc. Biological Sciences*, pp. 9-15, 1994.

[16] W. Maass, "Networks of spiking neurons: The third generation of neural network models", *Neural Networks,* vol. 10, pp. 1659-1671, 1997.

[17] W. Maass, "On the relevance of time in neural computation and learning", *Theoretical Computer Science, Elsevier,* vol. 261, pp. 157-178, 2001.

[18] S. Haykin, "Neural networks: a comprehensive foundation", *Prentice Hall,* 2008.

[19] S.M. Bohte, J.N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons", *Neurocomputing,* vol. 48, pp. 17-38, 2003.

[20] F. Ponulak, "ReSuMe-new supervised learning method for Spiking Neural Networks", *Technical Report, Institute of Control and Information Engineering, Poznan University of Technology,* 2005.

[21] W. Maass, "Fast Sigmoidal Networks via Spiking Neurons", *Neural Computation, MIT Press,* vol. 9, pp. 279-304, 1997.

[22] J.J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation", *Nature,* vol. 376, pp. 33-36, 1995.

[23] D. Meunier, and H. Paugam-Moisy, "A spiking Bidirectional Associative Memory for modeling intermodal priming", *International Conference on Neural Networks and Computational Intelligence,* pp. 25-30, 2004.

[24] N. Iannella, and A.D. Back, "A spiking neural network architecture for nonlinear function approximation", *Neural Networks, Elsevier,* vol. 14, pp. 933-939, 2001.

[25] W. Maass, and T. Natschlager, "Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding", *Network: Computation in Neural Systems,* vol. 8, pp. 355-371, 1997.

[26] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc. of the National Academy of Sciences,* vol. 81, pp. 3088-3092, 1984.

[27] J.A. Freeman, D.M. Skapura, "Neural networks: algorithms, applications, and programming techniques", *Addison Wesley Longman Publishing Co.,* CA, USA, 1991.

[28] B. Kosko, "Bidirectional associative memories", *IEEE Trans. Systems, Man and Cybernetics,* vol. 18, pp. 49-60, 1988.

[29] J.R. Koza, "Genetic programming: on the programming of computers by means of natural selection", *The MIT press,* 1992.

[30] J. Poon, and M.L. Maher, "Co-evolution and emergence in design", *Artificial Intelligence in Engineering*, vol. 11, pp. 319-327, 1997.

[31] W. Gerstner, and W.M. Kistler, "Spiking Neuron Models: Single Neurons, Populations, Plasticity", *Cambridge University Press*, 2002.

[32] R. Jolivet, and T.J. Lewis, and W. Gerstner, "The spike response model: a framework to predict neuronal spike trains", *Proc. Joint International Conference ICANN/ICONIP*, pp. 846-853, 2003.

[33] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing", *Neural networks*, vol. 14, pp. 715-725, 2001.

[34] W. Maass, G. Schnitger, and E.D. Sontag, "On the computational power of sigmoid versus boolean threshold circuits", *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pp. 767-776, 1991.

[35] A. Belatreche, L.P. Maguire, and M. McGinnity, "Advances in design and application of spiking neural networks", *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 11, pp. 239-248, 2007.