5-6-2012

Cellular Neural Networks with Switching Connections

Malcom Devoe
Georgia State University

Malcom W. Devoe Jr.
Georgia State University

CELLULAR NEURAL NETWORKS WITH SWITCHING CONNECTIONS

by

MALCOM DEVOE

Under the Direction of Dr. Igor Belykh

ABSTRACT

Artificial neural networks are widely used for parallel processing of data analysis and visual information. The most prominent example of artificial neural networks is a cellular neural network (CNN), composed from two-dimensional arrays of simple first-order dynamical systems ("cells") that are interconnected by wires. The information, to be processed by a CNN, represents the initial state of the network, and the parallel information processing is performed by converging to one of the stable spatial equilibrium states of the multi-stable CNN. This thesis studies a specific type of CNNs designed to perform the winner-take-all function of finding the largest among the n numbers, using the network dynamics. In a wider context, this amounts to automatically detecting a target spot in the given visual picture. The research, reported in this thesis, demonstrates that the addition of fast on-off switching (blinking) connections significantly improves the functionality of winner-take-all CNNs.

INDEX WORDS: Networks, Cellular neural network, Winner-Take-All, Blinking connections, Multi-stable system, Averaging

CELLULAR NEURAL NETWORKS WITH SWITCHING CONNECTIONS

by

MALCOM DEVOE

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2012

CELLULAR NEURAL NETWORKS WITH SWITCHING CONNECTIONS

by

MALCOM DEVOE

Committee Chair: Dr. Igor Belykh

Committee: Dr. Vladimir Bondarenko

Dr. Andrey Shilnikov

Dr. Michael Stewart

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2012

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support of all the people who lent me their supports in different ways. I would like to express my deepest gratitude and sincere appreciation to each and every of them. First of all, I would like to thank my advisor Dr. Igor Belykh for sharing his great wealth of knowledge in mathematics, especially in coupled dynamical systems with me. His encouragement, insightful guidance and his ongoing patience with me are highly appreciated and always remembered.

Second of all, I would also like to thank my committee members Dr. Vladimir Bondarenko, Audrey Shilnikov, and Dr. Michael Stewart for taking some valuable time out of their schedules to read my thesis and for giving me some critical suggestions in my thesis research.

I would also want to thank all the faculty members in the department of mathematics for teaching me and helping me develop my mathematical knowledge throughout my graduate study.

Thanks to all my classmates who helped me through these years and made it all bearable. Without the help of everyone, I would by no means be able to complete this research thesis.

This research project was supported under the NSF grant DMS-1009744.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 State of the Art, Motivation, and Goals

This thesis studies the advantages of information processing networks with fast on-off switching stochastic connections over the conventional networks with static structure in performing the "winner-take-all" function [1].

This research studies a class of stochastically switching networks, introduced by Belykh et al. [2] and called *the blinking model*. The blinking model represents a way of transforming a network with static connections into a small-world network with a time-varying structure.

Long-range small-world networks were proposed by Watts and Strogatz in a 1998 Nature paper [3], inspired by the small-world phenomenon (also known as six degrees of separation) observed in social networks. An example of a small-world network is a lattice of locally coupled cells that have a few, randomly chosen shortcuts. Small-world networks were then showed to significantly enhance propagation speed, information processing capabilities, and computation power due to the presence of small-world connections. Many real-world networks, including the Internet, electrical power grids, epidemiological and neuronal networks were showed to display the small-world structure (see [4] and the references therein).

In many engineering and biological networks, the small-world network structure changes as a function of time. The blinking model [2] introduced a time-varying small-world network by randomly choosing the shortcuts and leaving them fixed for a short interval of time τ, and choosing randomly choosing another set of shortcuts. More specifically, every possible shortcut is turned on with probability p, independently on switching of the other shortcuts, during each time interval τ. This switching time interval is assumed to be fast, compared to the dynamics of the

individual node, composing the network. Similar to the blinking of an eye, the connections rapidly turn on and off (see Fig. 1 (left panel)).

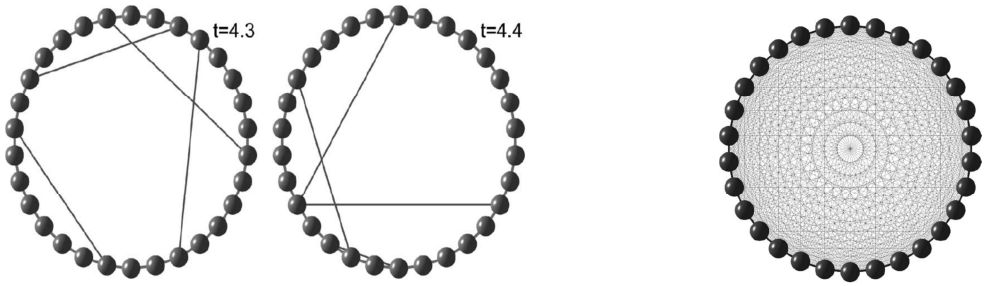

Fig. 1. [Modified from [2] for illustrative purposes. Courtesy of Dr. Igor Belykh]. The blinking model of shortcut connections. Probability of switchings $p = 0.01$, the switching time $\tau = 0.1$. The blinking model consists of the regular locally coupled lattice of 30 oscillators with constant coupling coefficients d and a time-dependent on–off coupling between any other pair of cells; when switched, the shortcuts have the same coupling strength d (left panel). Averaged network: the locally coupled lattice with the local coupling strength d and the additional global coupling pd. Here, p is small, such that the width of the links may be thought of as the coupling strength (a strong coupling within the local lattice and a weak coupling for the remaining all-to-all links) (right panel).

The blinking connections model realistic networks rather precisely. Examples of real-world networks with short on-off connections include packet switched networks such as the Internet. Neurons in the brain send out spikes and the neurons become effectively coupled during the short period of time when the spikes arrive at post-synaptic neurons. The simultaneous arrival of spikes to a given neuron in dense cortical networks, modeled by random networks, may be considered as a random process which represents blinking interaction of intermittent nature. Another important example of blinking interaction is synchronization of non-precise computer clocks by blinking network administration [2].

If the switching time τ is small then the dynamics of the blinking network can be similar to that of its averaged analog where the on-off stochastic connections are replaced with static global links as shown in Fig. 1 [2].

In [2,5-10], the relation between the dynamics of blinking networks and their averaged analogs was rigorously studied using the stability theory and averaging. It was shown [2,10] that the solutions of the blinking system converge to an attractor of the averaged system with high probability. In simple worlds, the averaged network describes the blinking stochastically switching network rather precisely, provided that the switching is fast compared to the intrinsic dynamics of each node. The fact that the rapidly switched system has the same behavior as the averaged system intuitively makes sense, but in fact there are exceptions, and therefore, a careful analysis of this property is needed which shows on what parameters the occurrence of the exceptions depends. This statement is made explicit in [2,10], and rigorous upper bounds linking the probability of converging to the same attractor, switching time, and intrinsic properties of the individual dynamical system are given.

In this thesis, the occurrence of the exceptions, that the multistable blinking and averaged networks converge to different attractors, will be studied in the context of information processing cellular networks. Such exceptions will represent the failure of the network to perform its function correctly.

The research objective of this thesis is to investigate how (i) the switching network topology and the properties of the individual nodes influence cooperative properties and the information processing capabilities of the blinking network and (ii) the addition of fast switching connections can enhance the performance of networks with static connections. Here, we exploit the above ideas of transforming local networks into small-worlds and study further the ad-

vantages of information processing CNNs with blinking connections over the conventional CNNs with static structure in performing the "winner-take-all" function [1,5].

More precisely, we study a cellular neural network (CNN), composed from two-dimensional arrays of simple first-order bistable dynamical systems that are interconnected by wires. Depending on the initial condition, each interacting cell converges to one of two equilibrium points, generating an output of +1 or -1. The information, to be processed by a CNN, represents the initial state of the network, and the parallel information processing is performed by converging to one of the stable spatial equilibrium states of the multistable CNN. This stable spatial equilibrium state is represented by the distribution of outputs +1 and -1.

In the following, we will study a specific type of CNNs designed to perform the winner-take-all function of finding the largest among the n numbers, using the network dynamics. One usually implements this by inserting data as initial values of the states and letting the states converge to an equilibrium point of the (multistable) network. The mapping from the initial to the final states is the function performed by the network. The result of the winner-take-all function is the convergence to an equilibrium spatial point where the cell with the largest initial value converges to the "+1" equilibrium points, whereas all the others cells with initial conditions, represented by smaller initial values, converge to the "-1" state. The "+1" winning cell represents the location of the largest number in the matrix. In a wider context, this amounts to automatically detecting a target spot in the given visual picture.

Unfortunately, this "winner-take-all" cannot be performed by a locally coupled CNN, that is very convenient for circuit implementation, and global connections are required. This point will be discussed in detail in Chapter 3. We use the stability conditions derived in [1] to design 4x4 and 10x10 CNNs with global static connections that reliably identify the largest

number (with 100% probability). However, hardwiring all-to-all connection in a large circuit is unrealistic. To resolve this issue, we will show that it is convenient to use a communication network, that is present to charge the initial conditions and read out the results, to establish on-off blinking connections that let the CNN perform the "winner-take-all" function correctly with high probability. In this setting, the CNN with global all-to-all static connections plays a role of the above averaged system for the blinking network (see Fig. 1 for the comparison). A rigorous upper bound on the probability that the multistable blinking CNN fails to converge to the correct spatial equilibrium and misclassifies the largest number was derived in [10]. In this thesis, we numerically verify this exponential dependence for the probably of an error on the negative reciprocal of the switching time τ.

These numerical studies required the development of MATLAB programs to run the extensive multi-hour simulations, especially in the case of 10x10 lattice with 100 nodes. These studies together with the efforts spent to get a deep insight into this new research field constitute the major part of the research performed in this thesis. Examples of the MATLAB programs are given in the appendix.

1.2 Thesis Outline

The outline of this thesis is as follows. In the next chapter (Chapter 2), we discuss the history and applications of conventional CNNs with local static connections. In Chapter 3, we introduce the models and study winner-take-all CNNs with (i) global static connections and (ii) switching blinking connections. Chapter 4 contains conclusions and discussions. The MATLAB codes are given in Appendix.

2. CONVENTIONAL CNN MODELS: HISTORY AND APPLICATIONS

2.1 Parallel Computing and Cellular Neural Networks

Parallel computing is the use of compute resources at the same time to solve computational problems. In other words, a problem is broken into parts that can be solved at the same time. For example, suppose there was a campaign manager who was in charge of advertising various flyers for promoting a mayor candidate. This manager has been given the task of the making 500,000 flyer copies that are to be delivered throughout the city. The task of creating these copies cannot be accomplished efficiently by the campaign manager himself; however, with the help of some 1000 team staffers who work in a building containing 1000 copiers, the job can be completed in less time than with campaign manager alone. If each staffer is position at a copier, then the job or task can be done 1000 times faster. This process of separating one complex job into several jobs to complete within a short amount of time is recognized as parallel computing. Parallel computing has been considered "the end of computing." Parallel computing has been used to solve difficult problems in many areas of science and engineering such as: Atmosphere, Earth Environment, Physics, Bioscience, Geology, Seismology, Mechanical Engineering, Circuit Design, Microelectronics, Computer Science, and Mathematics. The most common type of parallel computing is pipelining. With pipelining, the tasks are broken into steps performed by different units, with inputs streaming through, much like an assembly line. Parallel computing is also performed by means of artificial neural networks such as Cellular Neural Networks.

The Cellular Neural Network (CNN) is an artificial neural network that is represented by a collection of neurons that connected among each other; usually only local connections are chosen. The state of each cell is described mathematically by a dynamical system or a differential equation. The cells of the CNN network will only communicate with each other via sending sig-

nals to their neighboring cells. All cells in CNN have three main parts: the input coupling term, the state (cell), and an output coupling term. The condition of each cell relies heavily on the coupling terms from the input or output of its neighbor cells along with its initial condition. The CNN models are used in many real world applications such as analyzing 3D surfaces, solving partial different equations, and image processing. The CNN models can appear in many forms such as a ring, star, mesh, or a tree (see Fig. 2). The most popular form among the many different types is the eight-neighbor rectangular grid (see Fig. 3).

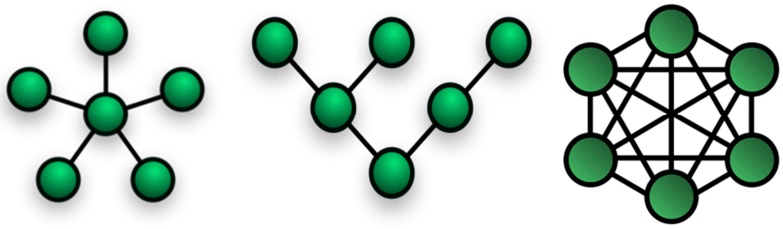

Fig. 2. Different CNN topologies (http://errajib.hubpages.com/hub/Types-of-Networks). (Left) Star network. (Middle) Tree. (Right) All-to-all global network. Each vertex is represented by a one-dimensional bistable dynamical system with two distinct outputs "+1" and "-1". The CNN system performs its information processing function by converging to a distribution of "+1" and "-1".

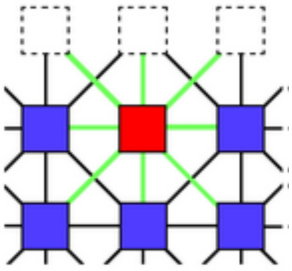

Fig. 3. The most popular CNN topology: eight-neighbor coupled network. Observe that three of its neighbors are boundary cells (dashed) [12].

The first cellular neural network was proposed by a Berkeley professor Leon Chua and his collaborator Lin Yang in 1988 [11,12]. This original CNN model, CY-CNN, used the weighted sum of the input and output to determine the condition or state at each cell. It is important to note that in a CNN model each cell is spaced equally among each like an *N* by *N* grid; however, the CNN model is not restricted to a two dimensional network. It also can be stretched to a finite N dimension of cells.

Today, many scientists develop CNN models to comprehend the biological settings that affect the environment, the human body, or the brain [13-19]. It is often used the show the responses of artificial intelligence. These models could be deterministic or stochastic depending on the dynamics or conditions of the environment. Through collecting data from an environment one is able to run experiments and develop a dynamical system or systems that satisfy the conditional of a single element. For instance, biologist and neuroscientist collect certain data from the brain to develop simple models that are coupled that describe mathematically how the brain sends signals from a single cell of the brain to another area.

2.2 Standard CNN equation: History

The general CNN model can be displayed as a system of nonlinear differential equations. We can use the basic first order cellular dynamics and interactions to describe the cell's state as follows:

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{(k,l)\in N(i,j)} A(i,j;k,l)y_{kl} + \sum_{(k,l)\in N(i,j)} B(i,j;k,l)u_{kl}, \tag{1}$$

$$y_{ij} = f\left(x_{ij}\right) = \begin{cases} 1 & \text{for } x_{ij} > 1 \\ x_{ij} & \text{for } -1 \le x_{ij} \le 1, \\ -1 & \text{for } x_{ij} < -1 \end{cases}$$

where $u_{ij}, x_{ij}, and\ y_{ij}$ are the input, the state, and the output of the cell in position *(i,j)*, respectively [12]. The indices *k* and *l* denote a cell that belongs to the neighborhood *N(i,j)*. Matrices *A* and *B* contain the weights of the neural network. The expression for the output y_{ij} is: $y_{ij}(t) = f\left(x_{ij}(t)\right) = \frac{1}{2}(|x_{ij}(t)+1| - |x_{ij}(t)-1|)$" (see Fig. 4). Given the input, the CNN performs its function by converging to a specific stable spatial equilibrium, corresponding to a distribution of the outputs -1 and +1 and reflecting the input signals. This point will be made clear in Chapter 3, discussing the Winner-take-all function performed by a CNN network.

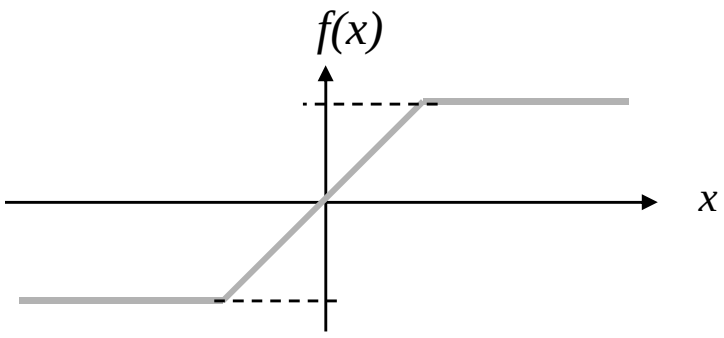

Fig. 4. Standard nonlinearity for the output equation in the CNN model (1).

Normally, the standard CNN model is created on an M ×N network of cells. When calculating the state of each cell, boundary conditions are a necessity to execute the model. The boundary conditions can be defined in several ways. The boundary conditions are able to be fixed where the value of the boundary cells is constant, zero-flux where the solution of the boundary cell matches the edge of cells, or periodic where the value of the boundary cells equals the value of the edge cells on the reverse side.

Figure 5 shows the topology of the standard *4x4* CNN model with r =1 where *r* represent the extent of the neighborhood. If *C(i,j)* is the cell on the i^{th} row and j^{th} column then cell *C(2,2)* is connected to *C(1,1), C(1,2), C(1,3), C(2,1), C(2,2), C(2,3), C(3,1), C(3,2),* and *C(3,3).* The r-

neighborhood is defined as: $N_r(i,j) = \{C(k,l)|max[|k-i|,|l-j|] \leq r, 1 \leq k \leq M; 1 \leq l \leq N\}$ with M and N the number of rows and columns respectively and r a positive integer.

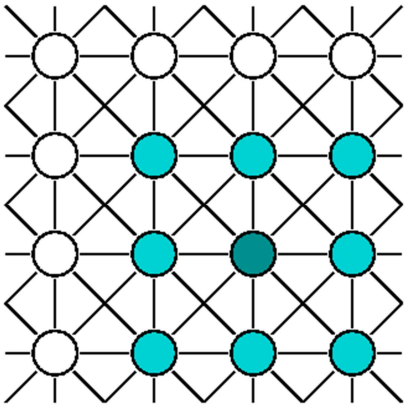

Fig. 5. A rectangular *4x4* grid CNN with a neighborhood radius of 1 [14].

The similar 8 × 8 grid CNN was called the CNN Universal Processor in 1993 [13]. This CNN Model has interfaces, analog memory, switching logic, and software. It was implemented to test the model's productivity and effectiveness. As a result in 2000, the usage of CNN models became very popular among many companies such as AnaFocus, a semiconductor company. The first CNN model that they created was called the ACE CNN processor. This ACE CNN processor had a 20 × 20 CNN processor unit. This model was later improved and lead to the development of an ACE processor that has 128 × 128 processor units. After rigorous developments of new CNN models to improve the performance of the previous model, AnaFocus found ways to increase the number of processing cells along with their speed and functional operations of each processing cells.

There are many advantages and disadvantages to the CNN model. The CNN model additional cells or neurons can be added to the network to extend the network. It can also perform tasks that a linear program cannot. When an element of the neural network fails, it can continue without any problem because of its parallel paradigm. Another advantage of the CNN model is that neural network can learn by adjusting its coupling strengths and does not need to be repro-

grammed. It can also be implemented in any application without any problem. The disadvantage of this model is that the neural network needs training to operate. The CNN requires high processing time for large neural networks.

We recall that the basic circuit unit of the CNN is called a cell. The cell holds linear and nonlinear circuit elements. These elements are normally linear capacitors, linear resistors, linear and nonlinear controlled sources, and independent sources. An illustration of a single cell circuit is shown in Fig. 6.

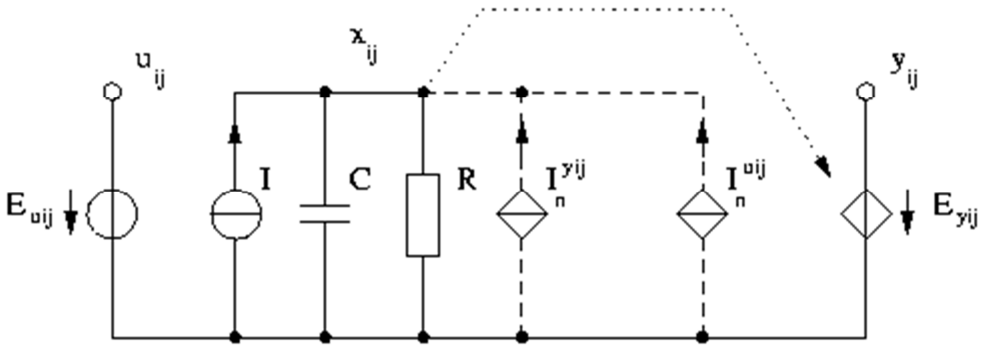

Fig. 6. [Picture taken from http://www.isiweb.ee.ethz.ch/haenggi/CNN_web/architecture.html]. Each cells has one independent voltage source E_{uij}, input, one independent current source I (bias), several voltage controlled current sources I_n^{uij}, I_n^{yij}, and one voltage controlled source E_{yij},(output). The controlled current sources I_n^{uij} are coupled to neighbor cells via the control input voltage of each neighbor cell. Similarly, the controlled current sources I_n^{yij} are coupled to their neighbor cells via the feedback from the output voltage of each neighbor cell.

Many scientists are motivated by the CNN models. Through studying the brain, scientists have found that the human brain is an extremely complex nonlinear system that consists of billions of simple processing elements, neurons. “Inspired by this biological network of neurons and deeply impressed by its signal processing capabilities, scientists and engineers design simpli-

fied artificial models with the far aim of achieving a performance comparable to the biological ideal [13]".

2.3 Applications of CNNs

CNN processors are used in many fields of science [13]. There are some applications that are engineering related, where some known, understood behavior of CNN processors is exploited to perform a specific task, and some are scientific, where CNN processors are used to explore new and different phenomenon [13]. CNN processors are used to do image processing; specifically, the first application of CNN processors was to perform real-time ultra-high frame-rate (>10,000 frame/s) processing with digital processors that are used in such as applications like particle detection in jet engine fluids and spark-plug detection. Currently, CNN processors are able to reach up to 50,000 frames per second. Applications such as missile tracking, flash detection, and spark-plug diagnostics are microprocessors that have surpass the performance of a conventional supercomputer. CNN processors are also used in local, low-level, processor intensive operations. "CNN processors have been used in feature extraction, level and gain adjustments, color constancy detection, contrast enhancement, deconvolution, image compression, motion estimation, image encoding, image decoding, image segmentation, orientation preference maps, pattern learning/recognition, multi-target tracking, image stabilization, resolution enhancement, image deformations and mapping, image inpainting, optical flow, contouring, moving object detection, axis of symmetry detection, and image fusion" [13].

CNN processors have exceptional processing capabilities and flexibility. They have been used or have been prototyped for applications such as flame analysis for monitoring combustion at a waste incinerator, mine-detection that uses infrared imagery, calorimeter cluster peak for physics, anomaly detection in potential field maps for geophysics, laser dot detection, metal inspec-

tion for identifying manufacturing defects, and seismic horizon picking. CNN processors have also been implemented to perform biometric functions like fingerprint recognition, vein feature extraction, face tracking, and generating visual stimuli through emergent patterns to gauge perceptual resonances. CNN processors have been made for medical and biological research to do automated nucleated cell counting to divide hyperplasia and segment images into anatomically and pathologically meaningful regions. The processors are great at measuring and quantifying cardiac function, measuring the timing of neurons, identifying brain abnormalities that would cause seizer activity. “One potential future application of CNN microprocessors is to combine them with the DNA microarrays to allow for a near-real time DNA analysis of hundreds of thousands of different DNA sequences. Currently, the major bottleneck of this DNA microarray analysis is the amount of time needed to process data in the form of images, and using a CNN microprocessor, researchers have reduced the amount of time needed to perform this calculation to 7ms” [13].

CNN processors have also been developed to create and analyze patterns and textures. One motivation was to use CNN processors to understand pattern generation in natural systems. Also, “CNN processors were used to approximate pattern generation systems that create stationary fronts, spatio-temporal patterns oscillating in time, hysteresis, memory, and heterogeneity Furthermore, pattern generation was used to aid high-performance image generation and compression via real-time generation of stochastic and coarse-grained biological patterns, texture boundary detection, and pattern and texture recognition and classification” [13].

Scientists are working to integrate CNN processors into sensory-computing actuating machines. This is done by creating an integrated system that uses CNN processors for the sensory signal processing and potentially the decision making and control. This is because CNN pro-

cessors yield a low power, small size, and eventually low cost computing and actuating system that is suited for that type of system. These Cellular Machines will eventually merge into a Sensor-Actuator Network (SAN), a Mobile Ad Hoc Networks (MANET) which is found in military intelligence gathering, surveillance of inhospitable environments, maintenance of large areas, planetary exploration, etc" [13].

CNN processors have also been proven versatile enough for some control functions [11]. "They have been used as associative memories, optimize function via genetic algorithm, measuring distances, optimal path finding in a complex, dynamic environment, and to learn and associate complex stimuli. CNN processors are used to design antonymous gaits by and low-level motor for robotic nematodes, spiders, and lamprey gaits using a Central Pattern Generator (CPG). "They [CNN processors] were able to function using only feedback from the environment, allowing for a robust, flexible, biologically inspired robot motor system. CNN-based systems were able to operate in different environments and still function if some of the processing units were disabled" [11].

The different types of dynamical behavior that are found in CNN processors make them interesting for communication systems. The turbulent communications that is used in CNN processors is being investigated because of their potential low power consumption, robustness and spread spectrum features. "The premise behind chaotic communication is to use a chaotic signal for the carrier way and to use chaotic phase synchronization to reconstruct the message." CNN processors are found in both the transmitter and receiver end to encrypt and decrypt the messages. They can also be made for source authentication through watermarking, detecting of complex patterns in spectrogram images (sound processing), and transient spectral signals detection" [11].

CNN processors are neuromorphic processors. This means that they are able to mimic certain aspects of biological neural networks. The first CNN processors were established on mammalian retinas, which are composed of a layer of photo detectors that were connected to many layers of locally coupled neurons. “This makes CNN processors part of an interdisciplinary research area whose goal is to design systems that leverage knowledge and ideas from neuroscience and contribute back via real-world validation of theories.” CNN processors have developed a real-time system that reduplicates mammalian retinas. This process validates that the original CNN architecture modeled the correct aspects of biological neural networks used to perform. “However, CNN processors are not only limited to verifying biological neural networks associated with vision processing; they have been used to simulate dynamic activity seen in mammalian neural networks found in the olfactory bulb and locust antennal lobe, responsible for preprocessing sensory information to detect differences in repeating patterns” [11].

CNN processors play a significant role in helping us understand systems that can be modeled living cells, biological networks, physiological systems, and ecosystems. The CNN architecture displays some of the dynamics that are observed in nature and is easy enough to analyze and conduct experiments. They are also used in stochastic simulation techniques. This allows scientists to venture spin problems, population dynamics modes, lattice gas models, and percolation. Some other simulations consist of heat transfer, mechanical vibrating systems, protein production, Josephson Transmission Line (JTL), seismic wave propagation, and geothermal structures. One particular CNN model, the 3D (Three Dimensional) CNN, has been invented in order to show that complex shapes are emergent phenomena to established a link between art, dynamical systems and VLSI technology. CNN processors are needed to study various mathematical concepts, such as analyzing non-equilibrium systems, building non-linear systems of arbitrary com-

plexity using a collection of simple, well-understood dynamic systems, investigating emergent chaotic dynamics, developing chaotic signals. The goal with the CNN model is to create a conceptual and mathematical framework necessary to analyze, model, and understand systems, including, but are not limited to, atomic, mechanical, molecular, chemical, biological, ecological, social and economic systems. “Topics explore are emergences, collective behavior, local activity and its impact on global behavior, and quantifying the complexity of an approximately spatially and topologically invariant system. Although another measure of complexity many not make some people enthusiastic (Seth Lloyd, a professor from Massachusetts Institute of Technology (MIT), has identified 32 different definitions of complexity), it can be potentially be mathematically analyze systems such as economic and social systems” [11].

2.4 Limitations of locally coupled CNNs: the need of global connections

The basic CNN model has many functions that can be computed by series of locally connected dynamical systems; however, many information processing functions require long range interactions of the cells for efficient computations. The fixation of all-to-all connections of *n* by *n* cells would require n^4 wires which is not realistic in most cases. A more effective approach to this is to develop an algorithm that shows that long distance connections can actually be switched on and off randomly in such a way that with high probability the computational function that the network performance is the same as that of a corresponding non-switched system, the averaged system.

In the next chapter, we will focus on a switching CNN that is capable of solving the winner-take-all function.

3. WINNER-TAKE-ALL CNNs

3.1 Conventional Model with Fixed Connections

3.1.1 Winner-take-all Model

We start off with conventional CNN model proposed by Seiler and Nossek in [1]. In this CNN model, each cell is self-connected and also connected to all other cells. Similarly to (1), the network dynamics can be described as the follows:

$$\frac{dx_i}{dt} = -x_i + \sum_{k=1}^{N} a_i^k y_k + \kappa, \tag{2}$$

$$y_i = f(x_i) = \begin{cases} 1, & x_i > 1 \\ x_i, & -1 \le x_i \le 1, \\ -1, & x_i < -1 \end{cases}$$

where the network consists of N all-to-all coupled cells. As in (1), x_i and y_i are the state and the output of the *i*-th cell. In contrast to (1), this network has no input variables u_i, and the input to the network is provided via the initial conditions of x_i. Parameter κ maintains a certain rate of convergence to a specific equilibrium point, and is present due to some historical reasons [1]. Parameter a_i^k is the coupling among cells. We assume that

$$a_i^k = \begin{cases} \alpha < 0, if\ i \neq k \\ \beta > 0, if\ i = k \end{cases}.$$

It is important to notice that $\alpha < 0$ and $\beta > 0$ so that the connections of a cell with the other neurons are *inhibitory* and self-connections are *excitatory* (Fig. 7). For convenience, we set the excitatory coupling strength $\beta = 1 + \alpha + \delta > 0$ with an auxiliary parameter δ chosen such that $\beta > 0$. Therefore, system (2) becomes:

$$\frac{dx_i}{dt_i} = -x_i - (\alpha y_1 + \alpha y_2 + \cdots + \beta y_i + \alpha y_{i+1} + \cdots + \alpha y_n) + \kappa,\ i = 1, \dots, N$$

and, consequently,

$$\frac{dx_i}{dt_i} = -x_i - (\alpha y_1 + \alpha y_2 + \cdots + (1 + \alpha + \delta) y_i + \alpha y_{i+1} + \cdots + \alpha y_n) + \kappa,$$

$i = 1, \ldots, N.$

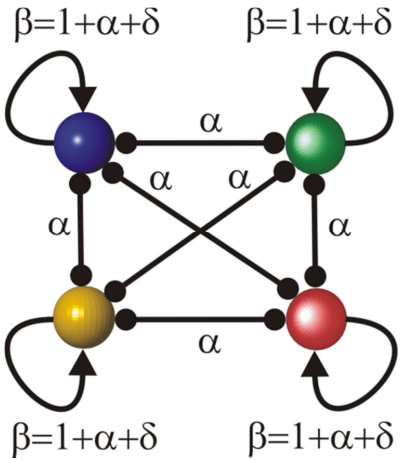

Fig.7. Four-cell network (2) of all-to-all connected cells with self-couplings. Intracellular connections are inhibitory (α<0). Self-connections are excitatory (***β***>0). The arrows indicate excitatory self-connections; the dots indicate mutual inhibition between the cells.

By separating the $(1 + \delta) y_i$ term from the summation we have the following system

$$\frac{dx_i}{dt} = -x_i + (1 + \delta) y_i + \alpha \sum_{i=1}^{N} y_i + \kappa, \text{ where } \alpha < 0 \text{ and } 1 + \alpha + \delta > 0. \qquad (3)$$

The excitatory self-connections with $\beta = 1 + \alpha + \delta > 0$ (see Fig. 7) are necessary for the CNN network (2) to perform the winner-take-all (WTA) function of finding the largest among the n numbers, using the network dynamics.

More specifically, the WTA function is performed as follows. The N given numbers are loaded to the network (2) as initial conditions $x_i(0)$, $i = 1, \ldots, N$, one for each cell. Let the largest

among these numbers be $x_m(0)$. Then, the vector of states ***x(t)*** evolves according to (2) and converges to the equilibrium point $\bar{\boldsymbol{x}}$ such that $\bar{x}_m \geq 1$ and $\bar{x}_i \leq -1$ for $i \neq m$. In terms of the outputs, this implies that $\bar{y}_m = f(\bar{x}_m) = 1$ and $\bar{y}_i = f(\bar{x}_i) = -1$ for $i \neq m$. Therefore, the dynamical system (2) must have N stable equilibrium points, one for each value for each *m*, corresponding to $x_m(0)$ with the largest initial value. The state space of multi-stable dynamical system (2) must be divided into the *N* attraction basins of the corresponding equilibrium points. The cell, that corresponds to the initial state with the largest value and converges to the +1 state, is called the "winning" cell; the other cells converging to -1 states are called "loosing" cells.

In other words, the WTA function is performed by network (3) if $\forall j \neq i$ if $x^j(0) < x^i(0)$ and for all initial conditions the eventual output of the "winning" cell (*i*-th cell) is +1 and that of the other cell -1. Below are the properties necessary for the WTA function.

Properties of all-to-all coupled WTA CNN (3):

To perform the WTA function, the following conditions must be satisfied:

1. $\frac{dx_i}{dt} < 0$ if the *i*-th cell is a "losing" one (non-winning)
2. $\frac{dx_i}{dt} > 0$ if the *i*-th cell is the "winner" where $\beta = 1 + \alpha + \delta > 0$.

This implies that state $x_m(t)$ with the largest initial condition must increase to reach the +1 state whereas the states of the other (loosing) cells must decrease to reach the -1 state. The excitatory connections are necessary for the winner cell to increase its state as they increase its time derivative. The inhibitory connections decrease the time derivatives of the loosing cells and force them to converge to the -1 states.

Theorem [Order Preserving Dynamics [1]]:

Consider CNN network (1). If the initial states of two cells i and j are ordered as follows

$$x^i(0) > x^j(0)$$

then for all times $x^i(t) > x^j(t)$. *As a result the arrangement of the cells by the level of their respective states stays consistent.*

Proof . To ensure that we do not lose the generality, we shall focus our discussion on cell 1 and cell 2 , and claim (to arrive at a contradiction) that $x^1(0) > x^2(0)$, and there exists is some T such that $x^1(T) < x^2(T)$. Since the states are continuous, then exists $0 < t_0 < T$ such that $x^1(t_0) = x^2(t_0)$. Using the principle of uniqueness of solutions and the interchanging of cell indices at t_0, it shows that $x^1(t) \equiv x^2(t)$. However, this contradicts the assumption that $x^1(0) > x^2(0)$ when $t = 0$. When this is applied to all pairs of cells, the arrangement of the cell described in the proposition is preserved. This completes the proof. □

Equilibrium Design

In order to construct a WTA network from *N*-cell network (3), we must make *N* WTA patterns stable, while making the other patterns unstable. We must also find the restricted values on the following parameters: $\alpha, \delta,$ and κ. Denote by *k* the number of + 1 states in the equilibrium pattern. Clearly, for the n-cell winner-take-all CNN (1) only P_1 must be stable as the other patterns become unstable:

$$P_k \text{ is } \begin{Bmatrix} stable & if\ k = 1 \\ unstable & if\ k \neq 1 \end{Bmatrix} .$$

This is illustrated in Fig. 8, using a four-cell CNN (3).

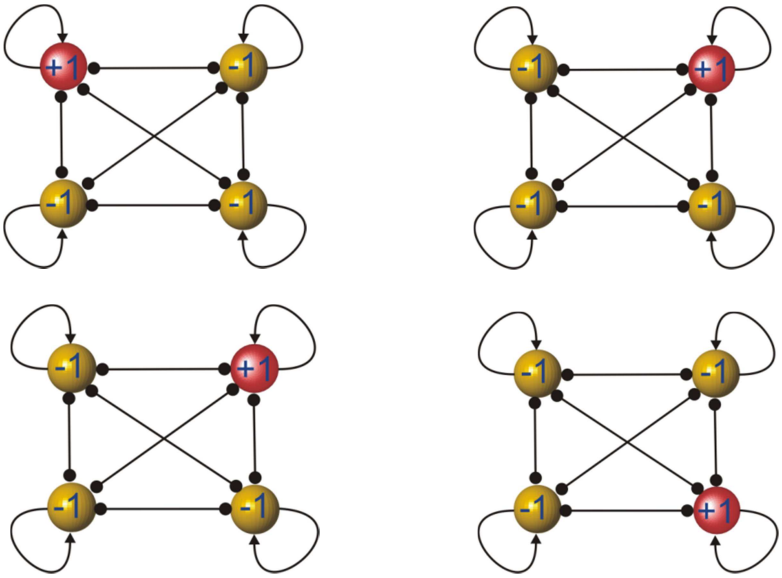

Fig. 8. Desired spatial equilibra of the WTA CNN. The outcome of the WTA function is the convergence to one of the four equlibria, depending on the initial conditions plugged in to the CNN. All other spatial equilibria (patterns), containing more than one winning cell (+1) must be made unstable by an appropriate choice of parameters $\boldsymbol{\alpha}, \boldsymbol{\delta}, \mathbf{and}\ \boldsymbol{\kappa}$.

We follow the steps of the previous study [1] to derive the stability conditions for the WTA patterns and the instability conditions for all other patterns with more than one winner.

First, we derive the conditions on parameters $\alpha, \delta,$ and κ that ensure the instability of the general pattern P_0 with *0* "winners" (+1 states) and *n* losers (-1 states). Similarly to Fig. 8, we call the winning cell a red one, and the loosing cells yellow ones.

If all states approach *-1*, then $\frac{dx_i}{dt} < 0$ for $i = 1,2,\ldots n$. For each of the *n* cells, we assume that we approach *-1* at all states. This yields the following equation for each cell:

$$\frac{dx}{dt} = -(-1) + (1+\delta)(-1) + \alpha(-n) + k < 0 \text{ where } i = 1,2,\ldots n. \quad (4)$$

Inequality (4) follows from system (3) where we have replaced all states x_i with -1 as we study the stability of all-loser pattern P_0. Solving (4) yields the following inequality

$$\kappa - \delta > n\alpha \tag{5}$$

The stability conditions of the WTA patterns P_1 where we have one winner and the rest of the cells are all losers are the following:

$$\frac{dx}{dt} = -(+1) + (1+\delta)(+1) + \alpha(1-1-1-1-1) + k > 0 \ (winner\ (red) cell)$$

or, in a shorter form

$$\frac{dx}{dt} = -(+1) + (1+\delta)(+1) + \alpha(-3) + k > 0 \ \ (winner\ (\ red) cell)$$

and similarly for the loosing cells

$$\frac{dx}{dt} = -(-1) + (1+\delta)(-1) + \alpha(-n) + k < 0 \ \ (losing\ (yellow)\ cell)$$

These two inequalities simplify to following forms:

$$\kappa - \delta < (n-2)\alpha \ \ \ (red\ cell) \tag{6}$$

$$\kappa + \delta > (n-2)\alpha \ \ \ (yellow\ cell)\ .$$

The rest of the patterns P_j with $j \geq 2$ such that there more than one winning cell (+1) in the pattern must be made unstable by choosing the parameters as follows. Generally, we have the choice of centering the instability condition on a winning or losing, but the WTA function requires the number of winning cells to be reduced to one cell. Therefore, the instability condition, that is developed from any winning (+1) cell is:

$$\forall 2 \leq j \leq N: \ \ \frac{dx}{d\tau} = -1 + \alpha(-n + 2j) + (\delta + 1) + \kappa < 0$$

which simplifies to

$\forall 2 \le j \le N: \qquad \delta + \kappa < \alpha(n-2j).$

When $j=2$, the inequality becomes $\delta + \kappa < \alpha(n-4)$. Out of all the inequalities, $\delta + \kappa < \alpha(n-4)$ is the most significant. By adding $\kappa - \delta > n\alpha$ with $-\kappa + \delta > -(n-2)\alpha$ and solving for α, we find that $\alpha < 0$.

When $j_1 < j_2$ with $\forall 2 \le j \le n: \delta + \kappa < \alpha(n-2j) \quad \alpha(n-2j_1) < \alpha(n-2j_2)$. If $j=2$ for $\delta + \kappa < \alpha(n-2j)$, then it will satisfy $\forall j \in \{\mathbb{N}\} - \{0,1\}$. Thus, the only inequality

$$\delta + \kappa < \alpha(n-4) \tag{7}$$

is sufficient to guarantee the instability of P_j.

Solving the inequalities (5), (6), and (7), we set $\alpha = -1$ and obtain the following sufficient conditions under which only the desired WTA patterns P_1 in CNN network (3) are stable[1]:

$$\alpha = -1, \qquad \kappa = -\frac{n^2-6}{n+2}, \qquad \delta = \frac{n+4}{n+2}. \tag{8}$$

Under these conditions, our CNN model will perform the WTA function and converges to one of the stable patterns P_1 with only one winner (+1).

In the following, we will consider two CNN networks (3) with 16 cells (4 x 4 network) and 100 cells (10x10 network). We shall use the stability conditions (8) to choose the parameters of the networks that guarantee the WTA function.

Condition (8), applied to a 16 cell network, yields

$$\alpha = -1 \text{ and } \kappa = -\frac{(16)^2 - 6}{16+2} = -13.89 \text{ and } \delta = \frac{16+4}{16+2} = 1.11 \tag{9}$$

For a 100-cell network we get:

$$\alpha = -1 \text{ and } \kappa = -\frac{(100)^2 - 6}{100 + 2} = -97.9804 \text{ and } \delta = \frac{100 + 4}{100 + 2} = 1.01961 \qquad (10)$$

Numerical examples, illustrating how the all-to-all connected CNN networks (3) perform the WTA function and always converge to the correct pattern P_1, therefore identifying the largest number, will be given in the next Section together with the comparison with small-world CNNs with sporadic connections.

3.2 Switching Small-World CNN Model

While the all-to-all connected CNN (3) reliably realizes the WTA function, its circuit implementation is not realistic in most case as hardwiring all-to-all connections of *N* cells would require N^2 wires. On the other hand, the conventional CNN circuits, used for information processing, usually consist of regular two-dimensional (2-D) arrays where next nearest neighbors are connected by wires (see the examples given in Chapter 1). The locally coupled CNN are very easy to implement; however, the WTA function cannot be obtained directly using only local connections. Here is a simple example illustrating this point. Assume that there are two local maxima in the initial state of a locally connected CNN, at cell *i* and cell *j* such that these maxima are sufficiently far away from each other. Assume that at cell *i* the maximum is global and therefore represents the largest number in the set of initial conditions. If this network performs the WTA function correctly, there must be a stable equilibrium for which the output of the *i*-th cell is +1 and all other outputs are −1. However, when all cells are in saturation, the *j*-th cell and *i*-th cell do not interact. Hence, there will be another stable equilibrium where, in addition to the *i*-th cell, the *j*-th cell has output +1, and again all other cells have output −1, representing a P_2 pattern, incompatible with the WTA function [5]. In simple words, the stability conditions, similar to (4)-(10), cannot be derived for the locally coupled CNN.

To resolve this dilemma of having an easily implemented CNN, that solves the WTA problem, we propose to use switched, instead of hardwired, global connections (shortcuts) and realize them by sending packets on a communication network that is associated with the CNN. This communication network is present anyway as one has to charge the initial conditions and read out the results. This CNN network with fixed local connections and on-off switching connections belongs to the class of blinking small-world networks discussed in Chapter 1 (see Fig. 1).

3.2.1 Model Equations

Similarly to the all-to-all connected CNN (1), we consider the following CNN with fixed nearest-neighbor connections and on-off switching shortcuts [5]. The equations read

$$\begin{aligned} \frac{dx_i}{dt} &= -x_i + (1+\delta)\, y_i + \alpha \sum_{\substack{j \text{ nearest} \\ \text{neighbor of } i}} y_j + \frac{\alpha}{p} \sum_{\substack{j \text{ not nearest} \\ \text{neighbor of } i}} s_{ij}(t)\, y_j + \kappa \\ y_i &= f(x_i) \end{aligned} \tag{11}$$

During each time interval $(k-1)\tau \le t \le k\tau$, the binary stochastic variable $s_{ij}(t)$ takes on value 1 with probability p and therefore activates a connection between cell i and cell j. Respectively, $s_{ij}(t)$ takes on value 0 with probability $1-p$ and switches off the connection between cell i and cell j if there was one, activated during the previous time interval. Therefore, we randomly choose the shortcuts and leave them fixed for a short interval of time τ, then we randomly choose another set of shortcuts (see Fig. 9). Every possible shortcut is turned on with probability p, independently on switching of the other shortcuts, during each time interval τ. Switching stochastic variables $s_{ij}(t)$ represent independently identically distributed (IID) sequences of a stochastic process. This switching time interval is assumed to be fast, compared to the convergence proper-

ties of the individual cell, composing the network. This justifies the name of the "blinking" CNN as the connections are switching on-off similarly to the blinking of an eye.

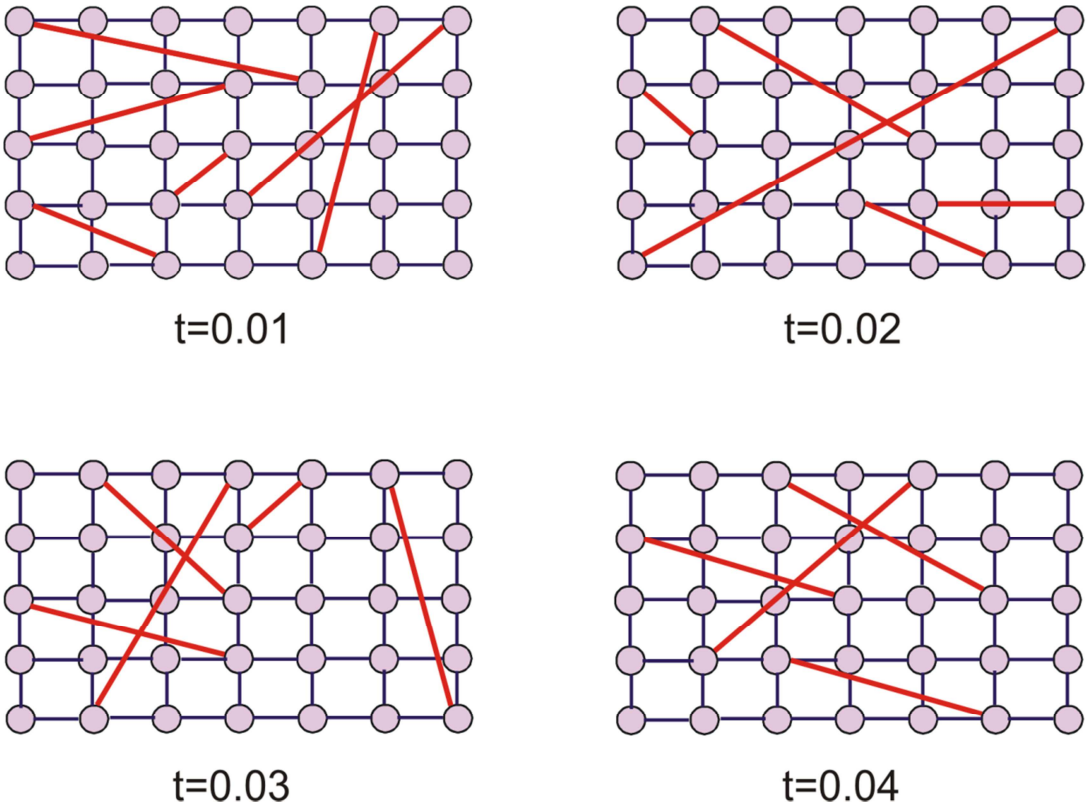

Fig. 9. Example of the CNN (11) with fixed local connections and on-off switching nonlocal connections. Probability of switching $p = 0.01$, the switching time $\tau = 0.01$. Given the probability, on average there are about 5 nonlocal connections activated during each time interval τ. This average number comes from the fact that there potentially are 537 nonlocal connections to switch and the probability of switching is $p = 0.01$, thus 537x0.01=5.37 connections.

If the connections in the blinking CNN model (11) switch fast enough, then its trajectory must follow closely the trajectory of the averaged system, obtained from the blinking system by replacing the binary stochastic signals $\boldsymbol{s}$ with constants p, where p is the switching probability [5,10]. It is important to stress that the all-to-all connected CNN (3) with fixed couplings is the

averaged system for the blinking CNN (11) where all stochastic variables $s_{ij}(t)$ are replaced with parameter *p*.

Recall that the all-to-all CNN (3) performs the WTA function reliably and finds the largest number with probability 1; however, its disadvantage is the necessity to wire all N^2 connections. The main idea of this thesis is to use the blinking CNN (11) as WTA CNN, instead of the all-to-all CNN with fixed connections. An advantage is the simplicity of its circuit implementation. A potential disadvantage is that the blinking (stochastic) CNN (3) doesn't always converges to the correct WTA pattern, causing the misclassification of the largest number, as switching is a stochastic process and the connections are not always present. However, if we switch fast enough, than the dynamics of the two CNNs (fixed and stochastic) become very similar, and the probability of misclassification tends to 0 as the switching time decreases. We study this relation in the next Subsections.

3.2.2 4x4 CNN: how fast the switching should be?

As a numerical example, let us consider a blinking 4 x 4 CNN (N = 16) (11) that has the task to find the largest among the 16 numbers given below

0.3244	0.3958	0.1871	0.2898
-0.3145	-0.2433	-0.1069	0.6359
0.1833	0.8200	0.1295	0.3205
-0.2983	0.7073	0.6433	-0.2161

(12)

We compare the performance of the 4x4 CNN (3) with fixed connections and the blinking CNN (11) in finding the largest number in matrix (12) (number 0.8200 in position (3,2)). To do so, we insert data from matrix (12) as initial values of the states x_{ij} so that number 0.3244 in position (1,1) is the initial value of the state x_{11}; number 0.3958 in position (1,2) is the initial value of the state x_{12} and so on. We then run the simulation of the CNN network (11) and letting

the states converge to an equilibrium pattern P_1 of the (multistable) network. Recall that there are 16 co-existing stable patterns P_1 with one "winner" number. The correct output of this simulation must be the convergence to the following pattern for cells' outputs y_{ij}:

-1	-1	-1	-1
-1	-1	-1	-1
-1	**+1**	-1	-1
-1	-1	-1	-1

Here, the location of +1 in the output of the blinking CNN indicates the location of the largest number in matrix (12).

Figures 10 and 11 show the simulations of the 4x4 blinking CNN (11) and 4x4 all-to-all connected CNN (3), that plays a role of the averaged system for the blinking CNN (the MATLAB codes are given in Appendix). It is demonstrated that while the all-to-all CNN always determines the largest number in position (3,2) of matrix (12), the performance of the blinking CNN depends on the generated stochastic sequence (a particular set of stochastically chosen and switched non-local connections) and on the switching time τ

Figure 10 demonstrates a specific instance of the blinking CNN model (11) for which the model determines the largest number correctly for the given switching time τ =0.001.

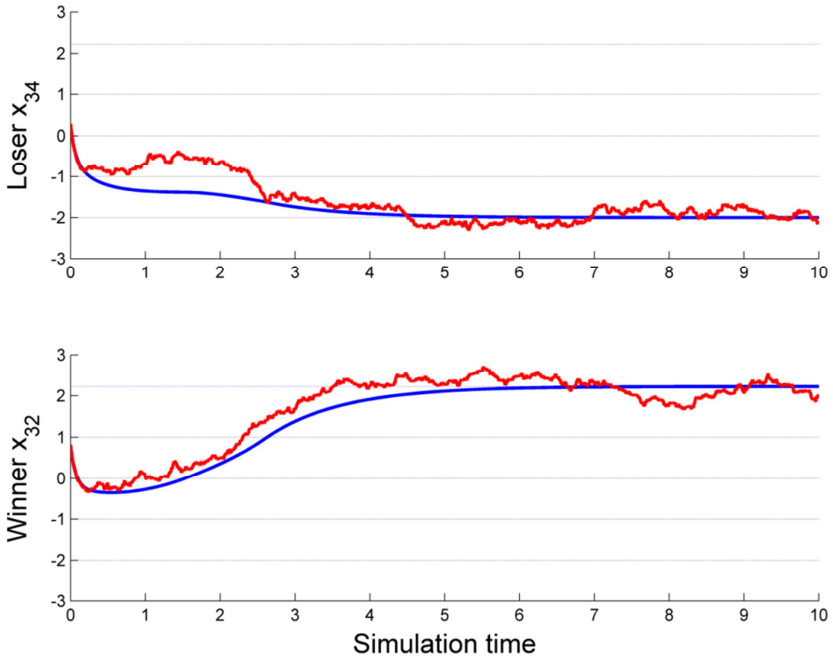

Fig. 10. Components $x_i(t)$ of trajectories for one instance of the 4×4 WTA blinking CNN (11) (irregular red lines) and for the 4×4 WTA CNN (3) with fixed all-to-all connections (smooth blue line). Parameters are calculated according to the stability condition (9): a = -1, δ =1.11, κ = −13.89. Switching time τ=0.001, probability of switching p = 0.1. (Top panel): The state $x_{(3,4)}$ of a losing cell (3,4), starts from a value that is lower than the largest number (cf. matrix (12)) and decreases below -1 as it should. All other losing cells have similar dynamics and converge to +1. (Bottom panel): The state $x_{(3,2)}$ of the winning cell (3,2), corresponding to the largest value 0.82, increases beyond the value +1, and therefore both CNNs identify the largest number correctly.

Figure 11 shows another instance of the blinking CNN model (11) for which the model fails to find the largest number. Note that the WTA CNN (3) with fixed all-to-all connections always determines the largest number with probability 1; however, it is cumbersome for circuit imple-mentation.

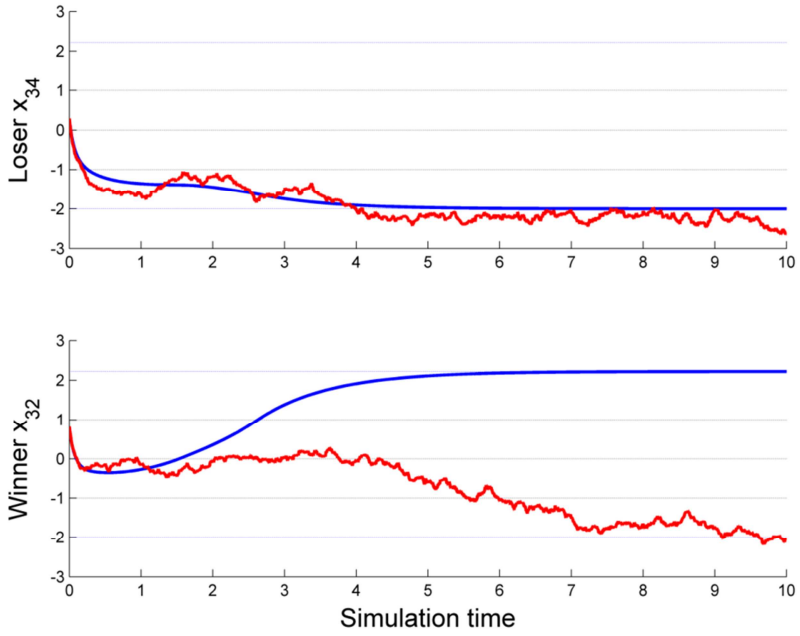

Fig. 11. Another instance of the 4×4 WTA blinking CNN (11) (irregular red lines) and for the 4×4 WTA CNN (3) with fixed all-to-all connections (smooth blue line). Parameters are the same as in Fig. 10. (Top panel): The state $x_{(3,4)}$ of a losing cell (3,4), starts from a value that is lower than the largest number (cf. matrix (12)) and decreases below -1 as it should. (Bottom panel): The state $x_{(3,2)}$ of the winning cell (3,2), corresponding to the largest value 0.82, increases beyond the value +1 in the CNN with fixed all-to-all connections as it should, but fails to do so in the blinking CNN. As a result, the blinking CNN misclassifies the largest number as cell (4,3) with the second largest number (0.6433) happens to reach the +1 state (not depicted in this Figure).

The main property of the CNN (3) with fixed all-to-all connections is that it has *N=4x4=16* co-existing stable patterns P_1 (stable spatial equilibrium points); however, these patterns are not equilibrium points of the blinking CNN (11). Therefore, as it is seen in Figs. 10-11, the trajectory of the blinking CNN cannot converge to an equilibrium point of the CNN with fixed all-to-all connections but can only approach it and wobble around. In practice, once the

trajectory of the blinking CNN gets sufficiently close to the desired stable equilibrium point of CNN (3) , one can classify the largest number and the system is stopped.

As shown in Fig. 11, there always is a non-zero probability of misclassification for the blinking CNN. However, it is relatively low and decreases when the switching time decreases. For the given switching time τ=0.001, we have run the simulations of the blinking CNN 100 times, starting from the same initial conditions given by matrix (12); there were 15 switching sequences out of 100 that lead to misclassification (one such sequence is depicted in Fig. 11). Our further numerical simulations showed that decreasing the switching time to τ=0.0001 reduces the probability of misclassification to P=2/100=0.02 as there were only 2 sequences, causing the convergence to the wrong attractor.

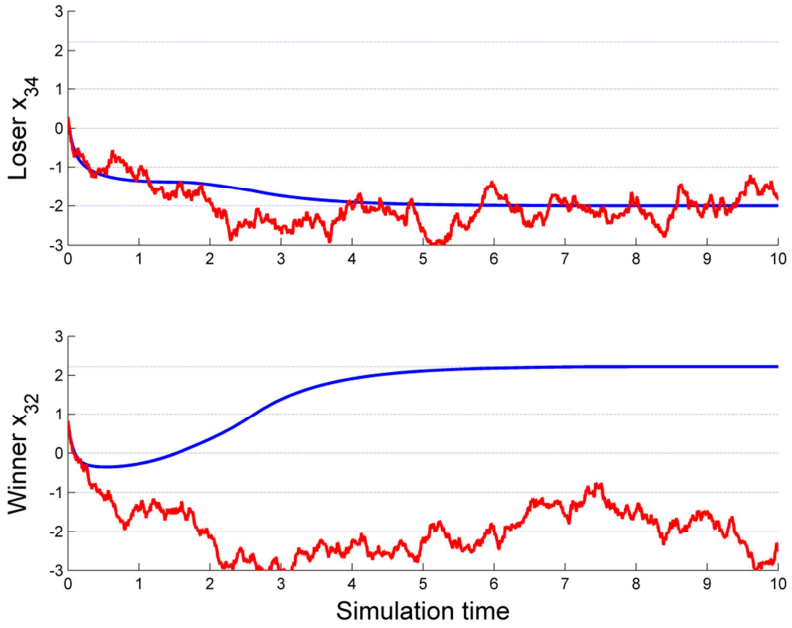

Fig. 12. Similar to Fig.11, except for different switching time τ=0.01. The given switching sequence leads to misclassification as the winner cell (3,2) converges to a wrong -1 state. The probability of

misclassification increases as the switching time increases. Note larger irregular oscillations of the blinking CNN (cf. Fig. 11); the switching is slower and the blinking CNN cannot stay sufficiently close to the CNN with fixed connections (blue smooth line).

In [5], Belykh *et al.*. used the Lyapunov function theory together with the averaging technique for stochastic differential equations to derive an upper bound on the dependence of probability of misclassification on the switching time. More specifically, it was shown that if the general multistable blinking dynamical systems and its averaged analog, where the switching parameters are replaced with their mean, start from the same initial condition and the averaged system converges to one attractor, then the probability that the blinking system doesn't converge to the same attractor, as it should and escapes to another attractor, tends to zero as the switching time approaches 0. Explicit bounds on this probability are given in [1,5,12]. In our context, the upper bound on the probability of escape in the blinking CNN, that causes misclassification, becomes [5]:

$$P_{misclass} \quad = \quad C_1 N^2 \exp\left\{-\frac{C_2 \gamma^3}{\tau}\right\},$$

where constants C_1 and C_2 are simple functions of parameters $\alpha, \delta,$ and κ of CNN model (3), parameter γ is defined by $\alpha, \delta,$ and κ and the initial condition chosen, and τ is the switching time as before. The actual formulas are tedious; however, their derivation from the general formulas, given in [5,12] is straightforward. Observe that the probability of misclassification

$$P_{misclass} \quad \sim \exp\left\{-\frac{1}{\tau}\right\} \tag{13}$$

can be made arbitrarily small by decreasing the switching time τ. However, estimate (13) comes sufficient conditions derived in [1,5,12]. In this thesis, we numerically verify this exponential dependence for the probably of an error on the negative reciprocal of the switching time

τ. As a result, we identify an optimal maximum switching time τ that keeps the probability of misclassification minimum. Evidently, to minimize this probability, one should decrease switching time τ, i.e. one should switch as fast as possible. However, faster switching results in high power consumption and, in addition, overload the communication network. Therefore, finding a trade-off between the switching time and the probability of misclassification is important. Fig. 13 demonstrates the results of multi-hour numerical calculations of the dependence of the probability of misclassification on the switching time (frequency). For each τ, we numerically integrate 4x4 CNN system (11), starting from the initial condition (12), and repeat the integration 100 times, counting the number of trials leading to misclassification for which cell 3,2 doesn't converge to the winner. This number divided by 100 trials gives us the probability of misclassification for the given each τ. Notice that due to the stochastic nature of switching, we have in principle 100 different stochastic sequences of switching for each τ. As shown in Fig. 13, the switching time smaller or equal τ=0.002, corresponding to the switching frequency $1/\tau = 500$ on the x axis of the graph in Fig. 13, gives an optimal bound for the switching time. Note that for the given switching frequency, the probability of misclassification drops dramatically and slowly decreases for values larger than 5000.

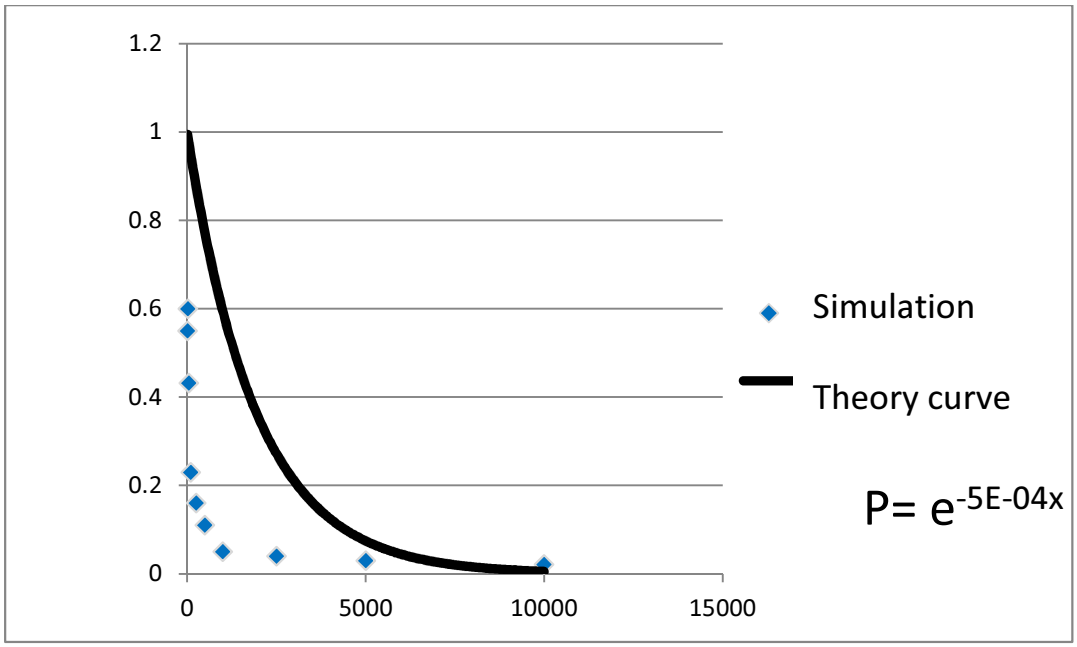

Fig. 13. Dependence of the probability of misclassification $P_{misclass}$ on the switching frequency $x = 1/\tau$. Each point (diamond) represents the results of 100 numerical solutions of 4x4 CNN system (11), starting from the same initial condition, but differing in the switching sequences, for each fixed switching frequency $1/\tau$. Switching frequency faster than 500 yields adequately low probability of misclassification. The solid line represents an exponential fit to the theoretical curve $P_{misclass} = \exp\{-1/\tau\}$.

3.2.3. 10x10 CNN: where is the spider?

In this subsection, we use a 10x10 CNN (11) to identify the darkest spot in a 2-D visual picture of Fig. 11.

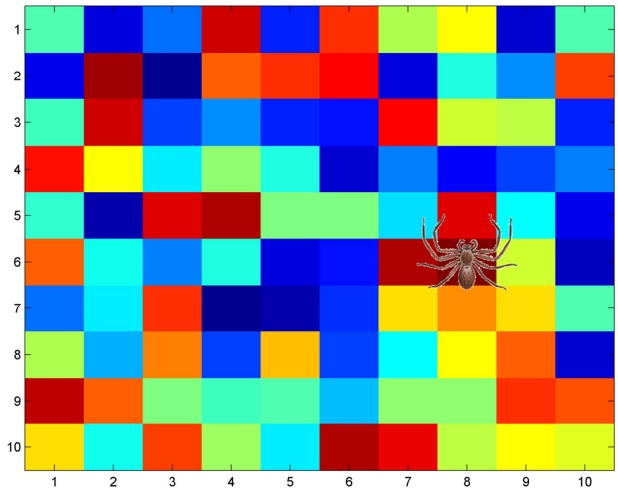

Fig. 14. 2-D picture with the darkest spot at cell 6,8 indicated by a spider. This picture is obtained from the below table using a Matlab command ‘image(A,'CDataMapping', 'scaled')’.

Table 1. 10x10 matrix with the largest number 0.9961 (cell 6,8) . This matrix is used as initial conditions for the 10x10 CNN (11). The CNN must perform the WTA function by converging to the pattern where cell 6,8 has an output +1 while the other cells converge to -1 states.

0.45	0.0838	0.2290	0.9133	0.1524	0.8258	0.5383	0.9561	0.0782	0.4427
0.1067	0.9619	0.0046	0.7749	0.8173	0.8687	0.0844	0.3998	0.2599	0.8001
0.4314	0.9106	0.1818	0.2638	0.1455	0.1361	0.8693	0.5797	0.5499	0.1450
0.8530	0.6221	0.3510	0.5132	0.4018	0.0760	0.2399	0.1233	0.1839	0.2400
0.4173	0.0497	0.9027	0.9448	0.4909	0.4893	0.3377	0.9001	0.3692	0.1112
0.7803	0.3897	0.2417	0.4039	0.0965	0.1320	0.9421	0.9961	0.5752	0.0598
0.2348	0.3532	0.8212	0.0154	0.0430	0.1690	0.6491	0.7317	0.6477	0.4509
0.5470	0.2963	0.7447	0.1890	0.6868	0.1835	0.3685	0.6256	0.7802	0.0811
0.9294	0.7757	0.4868	0.4359	0.4468	0.3063	0.5085	0.5107	0.8176	0.7948
0.6443	0.3786	0.8116	0.5328	0.3507	0.9390	0.8759	0.5501	0.6225	0.5870

Figure 15 demonstrates the results of numerical simulations and shows successful location of the largest number for the given switching sequence with the switching time τ=0.0001.

Similarly to the 4x4 CNN, we have calculated the probability of misclassification for 100 different switching sequences of the 10x10 CNN with the switching time τ=0.0001. While the probability of misclassification it is still acceptable (7/100), it's remarkably lower that the one of the 4x4 CNN with the same switching time. As a result, we come to a natural conclusion that while larger CNN networks (10x10 vs 4x4) give better resolution, the switching time τ must be

faster to maintain the same probability of classification (more cells, more co-existing winner-take-all patterns).

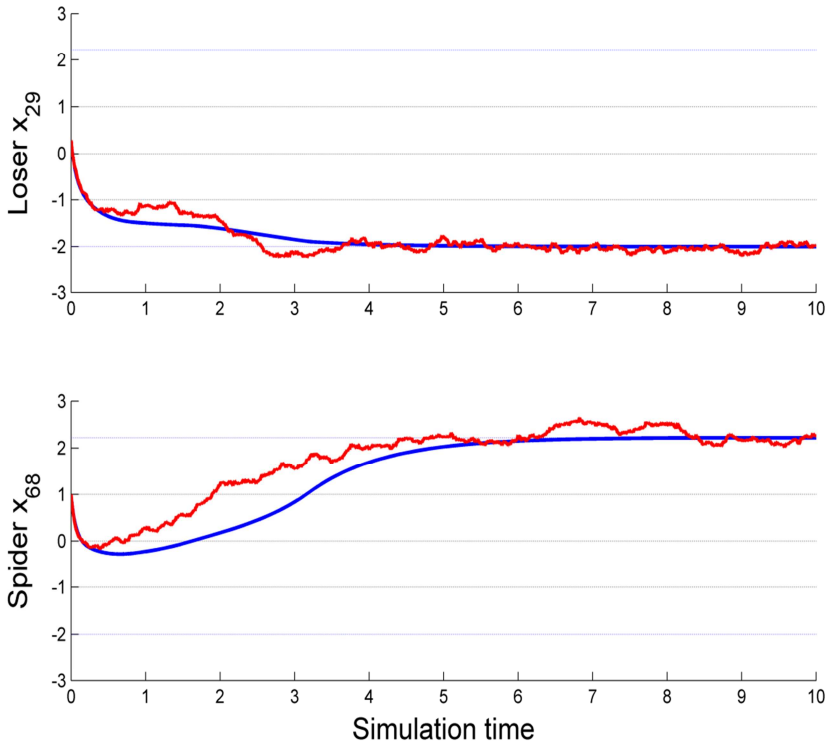

Fig. 15. Numerical simulations of a 10x10 all-to-all CNN (3) (blue smooth line) and a 10x10 switching CNN (11) (red irregular line) with parameters given in (10) and initial conditions from Table 1. The trajectory converges to the winner-take-all pattern: (top) a losing cell converges to a -1 state; (b) the winning cell, corresponding to the location of the spider in Fig. 14, converges to the +1 state. Switching time τ=0.0001. Probability of misclassification $P_{misclass} = 7/100$ (not depicted). Depicted is one of the successful 100-7=93 switching sequences that correctly identify the largest number (spider) in the matrix of Table 1 (image of Fig. 14).

4. CONCLUSIONS

We have analyzed one of the most prominent example of artificial neural networks such as a cellular neural network (CNN) and demonstrated that the addition of random on-off long-range connections significantly enhances functionality of locally coupled neural networks. In particular, we have studied the properties of winner-take-all (WTA) CNNs with on-off switching connections used to automatically identify the largest number in the given matrix. The WTA CNN performs parallel computation by using its cell dynamics when each cell of an N-cell network converges to either -1 or +1 state. The result is an equilibrium pattern, containing -1 and +1 states; for the problem in question this pattern is composed only of +1 "winner" state and *N-1* "losing" -1 states.

We have constructed WTA switching CNNs of different size (4x4 and 10x10 networks) and analyzed their performance for different switching frequencies. By performing extensive numerical simulations, we have shown that the probability of misclassification, for which the CNN fails to identify the largest number correctly, converges to zero exponentially fast as a function of the switching frequency. This allowed us to find an optimal switching frequency that yields a trade-off between the (low) probability of misclassification and the traffic load on the communication network used to establish fast stochastic on-off connections. We have also studied how the network size affects the probability of misclassification. More precisely, larger networks require faster switching to keep the same probability of misclassification as larger networks contain more cells and, therefore, have more WTA stable patterns.

REFERENCES

[1] Seiler, G. and Nossek, J., Winner-Take-All Cellular Neural Networks, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*., Vol. 40, pp. 184-190, 1993.

[2] Belykh, I., Belykh, V., and Hasler, M., Blinking model and synchronization in small-world networks with a time-varying coupling", *Physica D*, Vol. 195/1-2, pp 188-206, 2004.

[3] Watts, D. J. and Strogatz, S.H., "Collective dynamics of "small-world" networks", *Nature* 393, 440-442 (1998).

[4] Strogatz, S.H., "Exploring complex networks", Nature 410, 268-276 (2001).

[5] Belykh, I., and Hasler, M., Blinking Long-Range Connections Increase the Functionality of Locally Connected Networks, *IEICE TRANS. Fundamentals*, Vol.E88-A, pp. 2647-2655, 2005.

[6] Porfiri, M.M., Stilwell, D.J, Bollt, E.M., Skufca, J.D., Random Talk: Random Walk and Synchronizability in a Moving Neighborhood Network, *Physica D*, Vol. 224, pp. 102-113, 2006.

[7] Porfiri, M.M., Pigliacampo, R., Master-slave global stochastic synchronization of chaotic oscillators, SIAM J. Appl. Dynam. Sys., Vol. 7, 825-842, 2008.

[9] Hasler, M., Belykh, V., and Belykh, I., Dynamics of Stochastically Blinking Systems. Part I: Finite Time Properties. 2011 (submitted)

[10] Hasler, M., Belykh, V., and Belykh, I., Dynamics of Stochastically Blinking Systems. Part II: Finite Time Properties. 2011 (submitted).

[11] Chua, L. and Yang, L., Cellular Neural Networks: Theory, IEEE Trans. on Circuits and Systems, Vol. 35(10), pp. 1257-1272, 1988.

[12] Chua, L. and Yang, L., Cellular Neural Networks: Applications, IEEE Trans. on Circuits and Systems, Vol. 35(10), pp. 1273-1290, 1988.

[13] *Cellular Neural Network.* Wikipedia (The Free Encyclopedia), 2009.

[14] Hanggi, M. and Moschytz, G. S., Cellular Neural Networks: Analysis, Design, and Optimization., *Kluwer Academic Publishers*, 2000.

[15] Manganaro, G., Arena, P. , and Fortuna, L., Cellular Neural Networks: Chaos, Complexity, and VLSI Processing, *Springer*, 1999.

[16] Slavova, A., and Mladenov, V., Cellular Neural Networks: Theory and Applications. Nova *Science Publishers, Inc.* 2004

[17] Yang T., Cellular Neural Networks and Image Processing. *Science Publishers, Inc.* 2002

[18] Sum, John P. F., Leung, Chi-Sing, Tam, Peter K. S., Young Gilbert H., Kan, W.K., Chan, Lai-wan, Analysis for a Class of Winner-Take-All Model, *IEEE Transactions On Neural Networks*, Vol. 10, pp. 64-71, 1999.

[19] Feldman, J.A. and Ballard, D.H., Connectionist Models and Their Properties, Cognitive Science 6, pp. 205-254, 1982.

APPENDIX: MATLAB CODES

Matlab code:

```
% Main program to run
%
%
%
%
% Network size 10x10

nrows = 10;
ncols = 10;
n = nrows*ncols;

% Parameter design
alpha = 1;
kappa = -alpha*(n^2 - 6)/(n + 2);
delta = alpha*(n + 4)/(n + 2);
xeqp = delta + 1 + alpha*(n-2) + kappa;
xeqm = -delta - 1 + alpha*(n-2) + kappa;

A = diag(ones(n,1));

for i = 1:n
   [irow,icol] = ind2sub([nrows,ncols],i);
   for j = 1:i-1
      [jrow,jcol] = ind2sub([nrows,ncols],j);
      if abs(irow-jrow) + abs(icol-jcol) == 1
         A(i,j) = A(i,j) + 1;
      end
   end
end

plotprob=zeros(10,2);

% x0=0+(1-0).*rand(n,1);
%tau =.1;
increment=1;

%while (tau ~=.000001)

%x0 = 2*rand(n,1)-1;
%Initial conditions:
```

```
x0=[0.450541598502498;0.0838213779969326;0.228976968716819;0.913337361501670;0.152378018969223;0.825816977489547;0.538342435260057;0.996134716626886;0.0781755287531837;0.442678269775446;0.106652770180584;0.961898080855054;0.00463422413406744;0.774910464711502;0.817303220653433;0.868694705363510;0.0844358455109103;0.399782649098897;0.259870402850654;0.800068480224308;0.431413827463545;0.910647594429523;0.181847028302853;0.263802916521990;0.145538980384717;0.136068558708664;0.869292207640089;0.579704587365570;0.549860201836332;0.144954798223727;0.853031117721894;0.622055131485066;0.350952380892271;0.513249539867053;0.401808033751942;0.0759666916908419;0.239916153553658;0.123318934835166;0.183907788282417;0.239952525664903;0.417267069084370;0.0496544303257421;0.902716109915281;0.944787189721646;0.490864092468080;0.489252638400019;0.337719409821377;0.900053846417662;0.369246781120215;0.111202755293787;0.780252068321138;0.389738836961253;0.241691285913833;0.403912145588115;0.0964545251683886;0.131973292606335;0.942050590775485;0.956134540229802;0.575208595078466;0.0597795429471558;0.234779913372406;0.353158571222071;0.821194040197959;0.0154034376515551;0.0430238016578078;0.168990029462704;0.649115474956452;0.731722385658670;0.647745963136307;0.450923706430945;0.547008892286345;0.296320805607773;0.744692807074156;0.188955015032545;0.686775433365315;0.183511155737270;0.368484596490337;0.625618560729690;0.780227435151377;0.0811257688657853;0.929385970968730;0.775712678608402;0.486791632403172;0.435858588580919;0.446783749429806;0.306349472016557;0.508508655381127;0.510771564172110;0.817627708322262;0.794831416883453;0.644318130193692;0.378609382660268;0.811580458282477;0.532825588799455;0.350727103576883;0.939001561999887;0.875942811492984;0.550156342898422;0.622475086001228;0.587044704531417;]
    %x0=0+(1-0).*rand(n,1)

[xx0,ind] = sort(x0);

indmax = ind(n)
t0 = 0;
t1 = 10;
%tau = 0.0001;
%ntau = fix(t1/tau);

%Switching time;

tau =.0001;
ntau=100000;
t1 = tau*ntau;
t = [t0:tau:t1]';
tlength = length(t);
p = 0.1;
xx0 = x0;

% solution of the all-to-all CNN with fixed connections

[tf,xfull] = ode45('WTApwl',t,xx0,[],n,alpha,delta,kappa);
```

```
% solutions of the stochastic CNN

xblink = [x0'];

myprob=0;
for itau = 1:ntau
   B = rand(n,n) < p;
   AA = A + (1-A).*B*(1/p);
   [tb,xb] = ode45('WTApwl_var',[0 tau],xx0,[],n,AA,alpha,delta,kappa);
  % if(xb(itau)>1.5)
   %    myprob= myprob+1;
   %end
   nb = length(tb);
   xx0 = xb(nb,:)';
   xblink = [xblink; xb(nb,:)];

end
%finalprob=myprob/ntau
%xb(nb,:)

xblink1 = [x0'];
xx0 = x0;

for itau = 1:ntau
   B = rand(n,n) < p;
   AA = A + (1-A).*B*(1/p);
   [tb,xb] = ode23('WTApwl_var',[0 tau],xx0,[],n,AA,alpha,delta,kappa);
   nb = length(tb);
   xx0 = xb(nb,:)';
   xblink1 = [xblink1; xb(nb,:)];
end

if xblink(tlength,indmax) > 0
   attr = 1;
else
   attr = 0;
end


%plt = fix(rand*n)
plt = 4
%**************************************************************************
****take comments out and put the correct plots for figures 1 and 2 back from this sec-
tion**********************************
```

```
figure(1)
clf
subplot(2,1,1)
axis([0,10,-3,3])
hold on
plot([t0,t1],[0,0],'k:')
plot([t0,t1],[1,1],'k:')
plot([t0,t1],[-1,-1],'k:')
plot([t0,t1],[xeqm,xeqm],'b:')
plot([t0,t1],[xeqp,xeqp],'b:')
plot(t,xfull(:,plt),'b')
plot(t,xblink(:,plt),'r')
xblink(:,plt)

myprob=0;
myxblink=xblink(:,plt);
for i=1:101
   if(myxblink(i)>=1.5)
      myxblink(i)

      myprob=myprob+1;
   end
end
tau
myprob
final =myprob/101
plotprob(increment,:)=[tau,final]
increment=increment+1;
tau=tau/10;
%end

%*********************************************************************
*******
subplot(2,1,2)
axis([0,10,-3,3])
hold on
plot([t0,t1],[0,0],'k:')
plot([t0,t1],[1,1],'k:')
plot([t0,t1],[-1,-1],'k:')
plot([t0,t1],[xeqm,xeqm],'b:')
plot([t0,t1],[xeqp,xeqp],'b:')
plot(t,xfull(:,indmax),'b')
plot(t,xblink(:,indmax),'r')

%*********************************************************************
********
xeq = xeqm*ones(1,n);
xeq(indmax) = xeqp;
Mxeq = ones(tlength,1)*xeq;
mdevfull = sqrt(mean((xfull-Mxeq).^2,2));
```

```
mdevblink = sqrt(mean((xblink-Mxeq).^2,2));
if attr
   mdevblink = sqrt(mean((xblink-Mxeq).^2,2));
else
   mdevblink = sqrt(mean((xblink1-Mxeq).^2,2));
end

figure(2)
clf
semilogy(t,mdevfull,'b')
hold on
semilogy(t,mdevblink,'r')
%**********************************************************************
******
%**********************************************************************
******

%Function for the all-to-all fixed CNN*******************

function dx = WTApwl(t,x,init,n,alpha,delta,kappa)

% function WTApwl(t,x,init,n,alpha,delta,kappa)
dx = -x + (delta + 1)*fcnn(x) - alpha*ones(n,n)*fcnn(x) + kappa;


%Function for the switching CNN*******************

function dx = WTApwl_loc(t,x,init,n,A,alpha,delta,kappa)

dx = -x + (delta + 1)*fcnn(x) - alpha*A*fcnn(x) + kappa;



%**********************************************************************
******
%**********************************************************************
******

function y = fcnn(x)

% function y = fcnn(x)
% calculates the piecewise linear activation function used in CNN's.

y = -(x < -1) + (-1 <= x).*(x <= 1).*x + (1 < x);
```