CHAPTER 1

EMBEDDED SPIKING NEURAL NETWORK

1.1 INTRODUCTION

NEURAL networks are computational models of the brain. These networks are good at solving problems for which a solutions seems easy to obtain for the brain, but requires a lot of efforts using standard algorithmic techniques. Examples of such problems are pattern recognition, perception, generalization and non-linear control. In the brain, all communication between neurons occurs using action potentials or spikes. In classical neural models these individual spikes are averaged out in time and all interaction is identified by the mean firing rate of the neurons.

Recently there has been an increasing interest in more complex models, which take the individual spikes into account. This sudden interest is catalyzed by the fact that these more realistic models are very well suited for hardware implementations. In addition they are computationally stronger than classic neural networks.

Spiking neural networks are better suited for hardware implementations due to two facts: inter-neuron communication consists of single bits and the neurons themselves are actually only weighed leaky integrators. Because only single bits of information need to be transmitted, a single wire is sufficient for connection between two neurons. Thereby the routing of neural interconnection on a 2D chip is implied. The neural model that is used is the 'integrate-and-fire' model. Neurons here are leaky integrators, which fire and reset the neuron when a threshold is reached (fig.1).

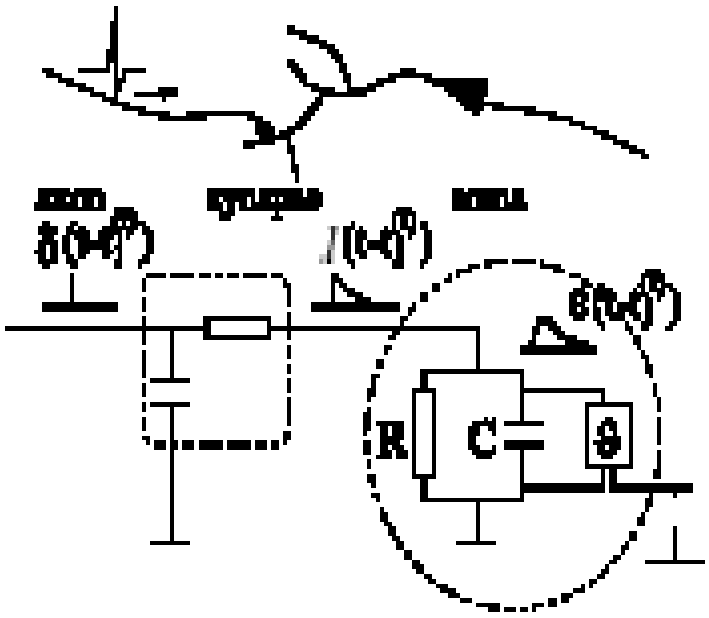

Fig. 1. An integrate-and-fire neuron.

Our research will primarily concentrate on digital implementations of these integrate-and-fire neurons. They can be implemented on standard reconfigurable hardware like Field Programmable Gate Arrays (FPGAs).

Now we will focus on the embedded side of the story. A current trend is to equip lots of devices with extra logic to make them interact more intelligently with their environment (ambient intelligence). This Intelligence is provided by an embedded system. Such an embedded system normally consists of a general purpose processor, some memory, interface hardware and some custom hardware executing very time critical tasks.

Recently the hardwired custom hardware gets replaced by a reconfigurable hardware component (fig. 2).

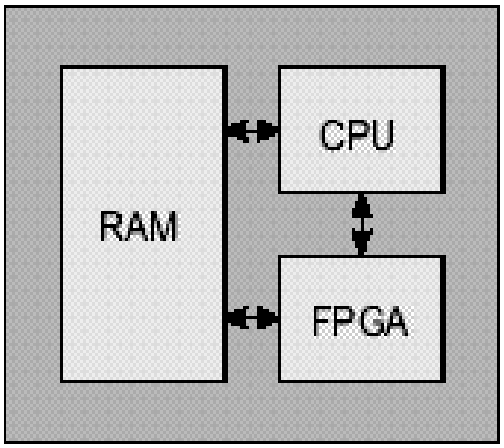

Fig. 2. An embedded architecture with an FPGA closely linked to the memory and CPU.

This way the function of that component can be redefined after design and manufacturing.

1.2 GOALS

This research will investigate if it is possible to implement spiking neural networks on re configurable hardware. More specifically it will concentrate on embedded devices because they have the advantage that a processor is closely linked to the neural module, which can coordinate learning, reconfiguration, etc. Furthermore it will research data representation, suitable spiking neural models, training, architecture and space/time-considerations.

Different spiking neural models will be investigated to see which are best suited for hardware implementation and is computationally the most interesting. Here both speed and area are important aspects.

To this date only very limited learning algorithms for spiking neural networks are available. This is mainly because this field of neural network research is pretty young. This research will address this problem.

Through the use of runtime re configurable hardware it is possible to split large networks into smaller sub modules, which can be implemented separately in hardware. Folding them out in time can thus run very large networks. This splitting of neural networks is actually a space versus time consideration.

Due to the fact that a processor is coupled with the neural hardware, it is possible to implement the additional learning phases (which only need to be performed sporadically during the lifetime of the neural module) on this processor. This way training can be done on a model in memory, and afterwards this model is translated to the actual hardware implementation.

1.3 PLANNING

The research will start by simulating many of the spiking neuron models present in the literature in software. This way it can compare the computational power, learning ability and complexity of these different models. During this stage research needs to be done on neural coding and training of these models because a comprehensive theory has not yet been formed.

In a second stage the best of these neuron models will be implemented in digital hardware (no complex connectivity will yet be implemented). In this stage we can evaluate speed and area requirements of these neuron models. We will use FPGAs to speed up simulation and evaluation significantly.

Then we will look at the interconnection of these neurons. Several different approaches are possible:

-Hardwired interconnections,

-Reconfigurable hardware connections,

-Message passing on a bus-structure, etc.

We will research how it is possible to implement a group of topologies as large as possible on the re configurable hardware without asking much resources, or introducing a big delay.

In the last step the best hardware neuron model and interconnection strategy are combined to form a neural hardware module. Research on reconfiguration bandwidth, memory architecture, pipelining, on-chip learning and space/time-considerations is needed. There will also
be special care in making the neural module as reusable as possible and therefore testability and the use of standard System-on-a-Chip busses will be considered .

As a result of this research, it will have a hardware neural network, implemental in digital re configurable hardware and suitable for embedded applications. It will be space/time-scalable, so that it can be easily adapted to the user requirements.

1.4 APPLICATIONS

All areas where embedded devices are used in complex changing environments, the neural network approach could amount for the adaptive and intelligent behavior. Due to the fact that the neural network is implemented in re configurable hardware it is possible to load the network only when it is needed.

It is now possible to solve complex problems in a hybrid way: partially by a neural network and partially by classic algorithmic techniques. Both

techniques can thus be used on the sub problems where they are best. Neural networks can for example be used to do pure pattern recognition on data, which is already preprocessed using classic techniques (for example Fourier transformation). The results of the neural network can then be fed to the processor, which can perform further computations.

A classic problem where we are faced with a complex, changing environment is biometrics. This term identifies all techniques used to measure human (or animal) features, for example voice, handwriting or iris recognition. All these human properties evolve during their lives and are very noisy. Hybrid techniques use neural networks to do low level pattern recognition (detecting single syllables) while algorithmic techniques are used in the higher levels (forming words and sentences with the syllables using dictionaries and statistical techniques) .

Another application is adaptive non-linear control. Most, if not all, machines are subject to significant wear and tear. Complex non-linear machines are locally linearised (in a working point) to enable control with standard techniques (like PID). Due to wear, the working point of the machine tends to shift so that an incorrect linear approximation is used and decent control gets impossible. Neural networks have proven to be very good at controlling highly non-linear systems.

Because these neural networks can be trained during operation, they can be adapted to changing situations like wear. Because this neural model can be implemented on an embedded device it is possible to place the control hardware very close to the detectors and actuators. This way delay (dead-time) is minimized, which normally is a big nuisance.

CHAPTER 2
SPIKES THAT COUNT: RETHINKING SPIKINESS IN NEURALLY EMBEDDED NETWORKS

2.1 INTRODUCTION

Models of spiking neurons have been extensively studied in the neuroscience literature, in recent years. Spiky networks have a greater computational power and are able to model the ability of biological neurons to convey information by the exact timing of an individual pulse, and not only by the frequency of the pulses.

This research will investigate the usage of spiking dynamics in embedded neurocontrollers that serve as the control mechanism for Evolved Autonomous Agents (EAAs) performing a delayed-response task. The spiky neural networks are developed by a genetic algorithm to maximize a behavioral performance measure, and their resulting networks and dynamics are subjected to further study.

EAAs are a very promising model for studying neural processing due to their simplicity, and their emergent architecture. Investigating spiky neural networks in this framework raises new questions that were not raised using pre-designed spiky models. For Example, evolutionary robotics studies have previously analyzed whether the spiking dynamics result in a time-dependent or a

rate-dependent computation, and investigated the effect of noise on the emerging networks.

It rigorously addresses the questions of what is a spiky network, and how to define and measure the spikiness level of each neuron, for the first time. We observe that a network with spiking neurons is not necessarily “spiky”, in terms of integration of inputs over time. Following this observation, we present two new fundamental ways by which we define and quantify the spikiness level of a neuron.

The study of spiking neural networks is performed within a delayed-response task, as memory is needed to solve such tasks and spiking dynamics may hence be useful. Delayed response tasks are characterized by a significant delay between the stimulus and the corresponding appropriate response, which make them impossible to solve by a simple sensory-motor mapping.

The rest of this is organized as follows:

Section 2 describes the network architecture and the evolutionary procedure. Section 3 we present two basic properties of spikiness in embedded agents. Section 4 analyzes the evolved neurocontrollers and their dynamics. These results and their implications are discussed in section 5.

2.2 THE MODEL

2.2.1 THE EAA ENVIRONMENT

The EAA environment is described in figure. The agents live in a discrete 2D grid world surrounded by walls. Poison items are scattered all around the world, while food items are scattered only in a food zone in one corner. The agent's goal is to find and eat as many food items as possible during its life, while avoiding the poison items. The fitness of the agent is proportional to the number of food

items minus the number of poison items it consumes. The agent is equipped with a set of sensors, motors, and a fully recurrent neurocontroller of binary neurons.

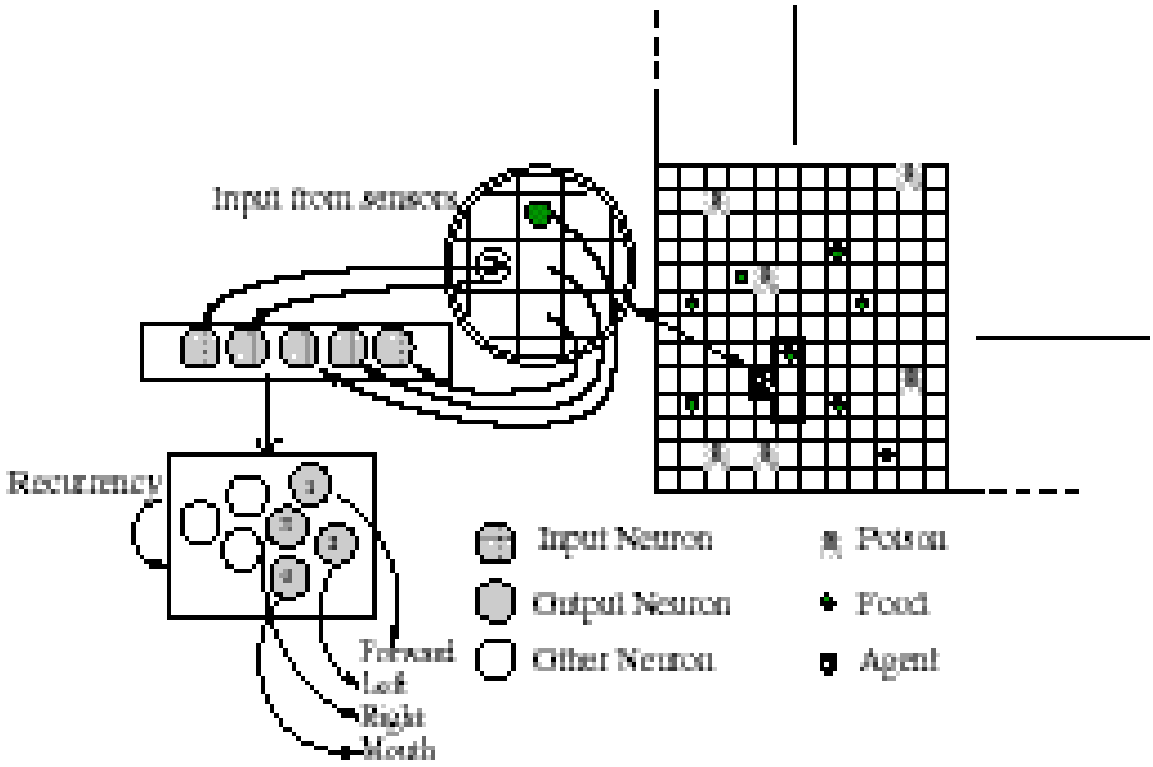

Figure 1: The EAA environment. An outline of the grid world and the agent's neurocontroller. A small arrow on the grid, whose direction indicates its orientation, marks the agent. The curved lines indicate where in the arena each of the sensory inputs comes from.

Four sensors encode the presence of a resource (food or poison, without distinction between the two), a wall, or a vacancy in the cell the agent occupies and in the three cells directly in front of it (Figure 1A). A fifth sensor is a smell sensor, which can differentiate between food and poison.

Underneath the agent, but gives a random reading if the agent is in an empty cell. The four motor neurons dictate movement forward (neuron 1), a turn left (neuron 2) or right (neuron 3), and control the state of the mouth (open or closed, neuron 4).

In previous studies, eating occurs if the agent stands on a grid cell containing a resource for one step. Here, we have modified this task to include a delayed-response challenge: In order to eat food, the agent has to stand on a grid-cell containing a resource for a precise number of steps K, without moving or turning, and then consume it, by closing its mouth on the last waiting step. Eating after standing on a food item for more or less than K steps does not increase its fitness.

Hence, in essence, the agent has to learn to count to K precisely. The agent's lifespan, defined by the number of sensor motor steps available to it, is limited. Waiting steps are not counted as part of lifespan steps in order to facilitate the evolution of the delayed-response task.

2.2.2 THE NEUROCONTROLLER

All neurocontrollers are fully recurrent with self-connections, containing 10 binary neurons (out of which 4 are motor neurons), and 5 sensor neurons that are connected to all network neurons. We compare between neurocontrollers with McCulloch-Pitts (MP) neurons, employed conventionally in most EAA studies, and ones with spiky Integrate-And-Fire neurons. In both types of networks,
a neuron fires if its voltage exceeds a threshold. The spiking dynamics of an Integrate-And-Fire neuron i are defined by,

$$V_i(t) = \lambda_i(V_i(t-1) - V_{rest}) + V_{rest} + \frac{1}{N}\sum_{j=1}^{N} A_j(t)W(j,i),$$

where Vi(t) is the voltage of neuron i at time t, ,i is a memory factor of neuron i (which stands for its membrane time-constant), Aj(t) is the activation (firing) of neuron j at time t, W(j; i) is the synaptic weight from neuron j to neuron i, N is the number of neurons including the input sensory neurons, and Vrest stands for the resting voltage (set to zero in all simulations). After ¯ring, the voltage of a spiky neuron is reset to the resting voltage, with no refractory period.

Evolution is conducted over a population of 100 agents for 30000 generations, starting from random neurocontrollers, using a mutation rate of 0.2 and uniform point-crossover with rate of 0.35.

2.3 PERFORMANCE EVALUATION

Successful agents that solve the delayed-response task were evolved with both MP and spiky networks. The evolution of the delayed-response task is fairly difficult, and many evolutionary runs ended (i.e. the performance has converged) without yielding successful agents. We measure the difficulty of each task as the fraction of runs that ended successfully. Evidently, the task is harder as the agent has to wait for a longer delay period. More important, successful spiky neurocontrollers evolve more easily than MP networks.

CONCLUSION

The study of spiky neural networks in the context of embedded evolutionary agents brings forward basic questions regarding spiking dynamics that have not yet been raised. The simplicity and concreteness of EAA models makes them a promising model for computational neuroscience research, and specifically to study the spikiness properties of neurocontrollers. Here it has shown that the presence of evolved spiking dynamics does not necessarily transcribe to actual spikiness in the network, and that the spikiness level can be defined and quantified in several functionally different ways. On a behavioral level it has shown that in tasks possessing memory-dependent dynamics network solutions that involve spiking neurons can be less complex and easier to evolve, compared with MP networks.

BIBILIOGRAPHY

[1] Embedded Spiking Neural Network
Benjamin Schauwen

[2] Spikes that count: rethinking spikiness in neurally embedded systems
Keren Saggie, Alon Keinan, Eytan Ruppin
School of Computer Sciences,, Tel-Aviv University, Tel-Aviv, Israel
{keren,**keinan}@cns.tau.ac.il,ruppin@post.tau.ac.il**
School of Medicine, Tel-Aviv University,Tel-Aviv,Israel

APPENDIX

–