ELectronics EXpress

IEICE Electronics Express, Vol.5, No.22, 921–926

Self-configuring spiking neural networks

Jose L. Rosselló[a)], **Ivan de Paúl, and Vincent Canals**

Electronics Systems Group, Balearic Islands University

Cra. Valldemossa km. 7.5, Palma de Mallorca 07122, Balearic Islands, Spain

a) *j.rossello@uib.es*

Abstract: We present a simple architecture for Spiking Neural Networks self-configuration. It consists in the hardware implementation of a simple Genetic Algorithm that may be used to obtain optimum network configurations. The proposed solution is applied to estimate the processing efficiency of different networks. Based on the results we develop a new performance metric to calibrate the processing capacity of SNNs.

Keywords: neural networks, spiking neural networks, hardware implementation of genetic algorithms

Classification: Photonics devices, circuits, and systems

References

[1] R. Malaka and S. Buck, "Solving nonlinear optimization problems using networks of spiking neurons," *Proc. Int. Joint Conf. on Neural Networks*, vol. 6, pp. 486–491, 2000.

[2] D. M. Sala and K. J. Cios, "Solving graph algorithms with networks of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 10, no. 4, pp. 953–957, July 1999.

[3] W. Gerstner and W. M. Kistler, *Spiking neuron models*, Cambridge University Press, 2002.

[4] N. Langlois, P. Miche, and A. Bensrhair, "Analogue circuits of a learning spiking neuron model," *Proc. Int. Joint Conf. on Neural Networks*, vol. 4, pp. 485–489, 2000.

[5] T. Dowrick, S. Hall, L. McLaid, U. Buiu, and P. Kelly, "A biologically plausible neuron circuit," *Proc. Int. Conf. on Neural Net.*, Orlando, USA, pp. 715–719, 2007.

[6] M. J. Pearson, C. Melhuish, A. G. Pipe, M. Nibouche, I. Gilhesphy, K. Gurney, and B. Mitchinson, "Design and FPGA implementation of an embedded real-time biologically plausible spiking neural network processor," *Proc. Int. Conf. on Field Programmable App.*, pp. 582–585, 2005.

[7] H. Torikai, A. Funew, and T. Saito, "Aproximation of spike-trains by digital spiking neuron," *Proc. of Int. Joint Conf. on Neural Net.*, Orlando, USA, pp. 2677–2682, 2007.

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008

1 Introduction

Practical hardware implementation of large neural systems is one of the major challenges for science and technology. Due to their parallel-processing nature, among other applications, neural systems can be used for real-time pattern recognition tasks and to provide quick solutions for complex problems that are intractable using traditional digital processors [1, 2]. The distributed information processing of Neural Networks also enhances fault tolerance and noise immunity with respect to traditional sequential processing machines. Although all these processing advantages, one of the main problems of dealing with neural systems is the achievement of optimal network configurations since the network complexity increases exponentially with the total number of neuron connections. Therefore, the development of learning strategies to quickly obtain optimum solutions when dealing with huge network configuration spaces is of high interest for the research community. The use of useful metrics to measure the network processing capacity is also of high importance. Therefore, a generic network processing metric would be ideal to compare different types of neural architectures or learning strategies.

Recently, a lot of research has been focused on the development of Spiking Neural Networks (SNN) [3] as they are closely related to real biological systems. In SNN information is codified in the form of voltage pulses called Action Potentials (APs). At each neuron cell the AP inputs are weighted and integrated in a single variable defined as the Post-Synaptic-Potential (PSP). The PSP is time dependent and decays when no APs are received. When input spikes excite the PSP of a neuron sufficiently so that it is over a certain threshold, an Action Potential is emitted by the neuron and transmitted to the rest of the network.

Different methodologies have been developed to implement SNNs using analogue [4, 5] and digital circuitry. Digital solutions [6, 7] provide lower design costs, and simpler configurability while the total design cycle is shorter since digital systems are easily configured and modified using FPGA platforms.

In this work we present a digital implementation of SNN. We also develop a simple architecture that can be used for SNN self-configuration. The proposed self-learning solution has been applied for temporal pattern recognition. We also propose a new generic metric to estimate the processing capacity of neural systems. The rest of this paper is organized as follows: in section 2 we show the proposed SNN self-learning architecture, while in section 3 we apply the proposed system to temporal pattern recognition analysis. We also present a new metric to estimate the network processing capacity. Finally in section 4 we present the conclusions.

2 Digital SNN architecture

As mentioned in the previous section, in SNN the information is codified in the form of pulses. We used a simplified digital implementation of the real behavior of biological neurons. In the proposed system, the PSP decay after

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008

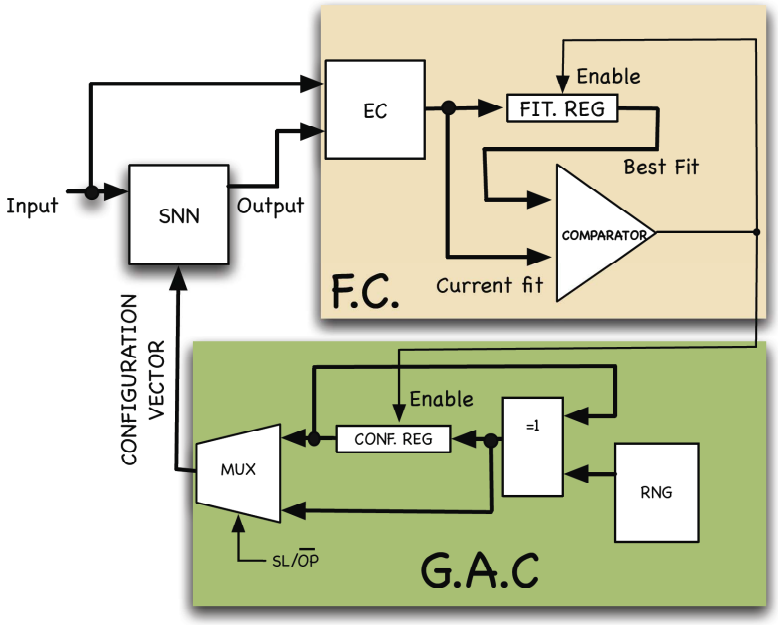

Fig. 1. Block structure for the dynamic configuration of SNN. The GAC block is used for the network configuration while the FC block evaluates the network efficiency.

each input spike is selected to be linear instead of the real non-linear variation while the refractory period present after each spike emission has been neglected. The main objective of this work is not to provide an exact copy of the real behavior of biological systems but to develop a useful Neural Network configuration technique and a generic performance estimation metric.

In the digital version implemented the PSP is codified as a digital number. At each spike integration the PSP is increased a fixed value that depends on the type and the strength of the connection. Therefore, positive (negative) increment values are associated to excitatory (inhibitory) connections. Each neuron is implemented using a VHDL code in which the connection strength is selected to be a fraction of the neuron threshold (in particular, these fractions are selected to be $\pm 2/5$ and $\pm 1/10$).

The proposed self-learning architecture is very simple as shown in Fig. 1. It consists in two basic blocks, a Genetic Algorithm Circuitry (GAC) and a Fitness Circuitry (FC). The GAC generates new configurations based on the better configuration obtained, that is stored in the configuration register. Using a Random Number Generator (RNG) (in this work we used an LFSR digital circuit) a random mutation vector is generated. The mutation vector is operated using XOR gates with the better configuration found until the moment (placed in the configuration register). The result is a new configuration (binary output of XOR block) that is equal to the previous except in those cases where the RNG provides a HIGH state. The new configuration is directly applied to the SNN when the controlling signal of the GAC multiplexer (SL) is HIGH (self-learning selection). When signal SL is LOW (operation mode) the better configuration obtained until that moment

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008

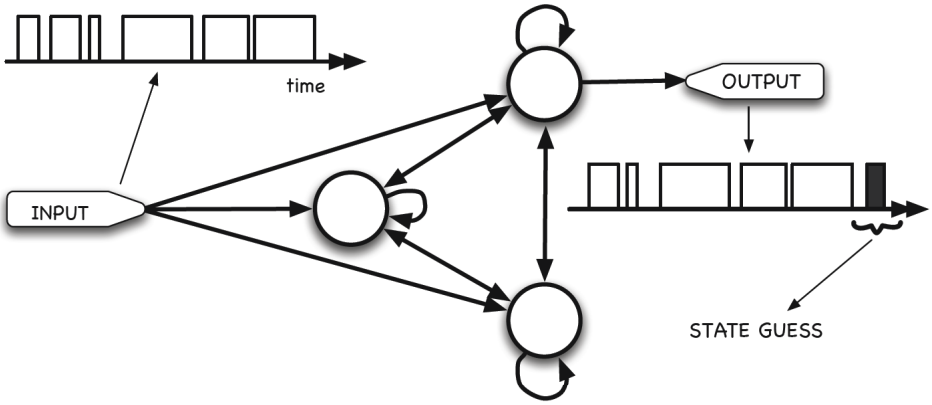

Fig. 2. A complete SNN implements all the possible inter-neuron connections. Such networks are trained to recognize temporal patterns

is applied to the SNN.

The FC block evaluates the aptness of each new configuration for a selected network task. During the training mode an evaluation circuitry (EC) compares the SNN behavior with the expected behavior, thus evaluating a cost function (the configuration fitness). The value obtained in this process is then compared to the one associated to the better configuration at the moment (stored at the fitness register). When a better fitness is found, the digital comparator output is set to a HIGH state and both the fitness and the configuration registers are updated with the new configurations at the end of the evaluation time.

When the system is in operation mode, the SNN configuration is fixed to the better solution obtained at the moment. A global reset is used to start with pre-selected initial conditions.

3 Neural processing capacity: the M-index

We applied the proposed SNN architecture to evaluate the processing behavior of various networks. The selected SNN task is the temporal pattern recognition (that is directly related to the "memory" capacity of the system). During the training mode, a finite sequence of vectors is applied repeatedly at the SNN input (the training bit sequence) and the task of the network consists in recognizing the sequence. At each time step the SNN has to provide the next bit of the sequence (see the illustration of Fig. 2). The network efficiency is evaluated estimating the probability of the SNN prediction success. At each evaluation step a mutated configuration provided by the GAC is used to configure de SNN. The FC bloc evaluates the probability of success of SNN predictions and the mutated configuration is therefore stored or discarded (see Fig. 1).

We configured different networks containing three, four and five neurons, each one connected to the rest of the network thus assembling a complete topology. In Fig. 2 we show the case with three neurons (defined as a 3-SNN).

The training bit sequence must be as complex as possible to maximize

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008

Table I. Temporal recognition effectiveness of SNN (success probability)

N	3-SNN	Model	4-SNN	Model	5-SNN	Model
7	86	94	100	99	100	100
15	80	80	80	83	87	87
31	71	71	74	73	74	75
63	67	65	67	66	67	68
127	62	60	62	62	62	63
255	59	57	59	58	60	59
511	56	55	57	56	56	56
1023	54	54	54	54	54	54

the pattern recognition difficulty. Therefore, we selected the generation of pseudo-random strings provided by LFSR digital blocks. Pseudo-random bit strings are characterized to have the same statistical properties as random sequences with the only characteristic that pseudo-random sequences have a periodicity. As is known, the periodicity of an LFSR constructed with 'n' registers is equal to $N = 2^n - 1$. In our experiment we used training sequences of N = 7, 15, 31, 63, 127, 255, 511 and 1023 bits. With the selection of this type of pseudo-random sequences, the memorization task difficulty is maximized.

We applied each selected pseudo-random training sequence to three different SNN with complete topology (using 3, 4 and 5 neurons). Each SNN is configured by the proposed genetic-based self-learning architecture. At each time step, the network has to guess the next bit that will be provided by the LFSR. Once the configuring circuitry has been stabilized to an optimum configuration we evaluate the probability of success associated to this final configuration. In Table I we provide the different prediction success (in percentage) for each network and training sequence together with the results of the model explained next. It is observed that, as parameter N decreases and as the number of neurons increases the network has a higher prediction success.

From measurement data, we inferred a rule to estimate the prediction success of each network. This rule is found to provide very close results to the measured data as is shown in Table I. The rule is an analytical expression that relates the success probability (p) to the training sequence length (N):

$$p = \frac{1}{2} + \frac{2}{5}\sqrt{\frac{M}{N}} \tag{1}$$

where parameter M is dependent on the type of the network (number of neurons, connection topology, etc.). For the networks trained we obtained M=8.5, 10.5 and 12.5 for the 3-SNN, 4-SNN and 5-SNN networks respectively. The expression in Eq. (1), indicates that as the length of the pseudo-random sequence increases the success probability of the network decreases to the limit value of 0.5 (50% of success probability).

The Parameter M (that we refer to as the network M-index) measures

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008

the ability of the system to recognize a pseudo-random bit sequence and can be used to estimate the processing potential of the system. The SNN M-index is defined as *the maximum pseudo-random sequence length that the network is able to recognize with a 90% success probability.* This index is a good indicator of the network processing capability, independently on the type and topology of the selected neural system.

In Table I we compare the predictions of Eq. (1) with the measured data showing a close relationship between measurements and the model developed. The major discrepancy occurs for the 3-SNN when N = 7. This is due to the fact that our model is continuous while the obtained success probability is discrete. That is, the prediction success for N = 7 is always a rational number with the denominator equal to 7. Therefore, the prediction of the continuous model (a 94%) cannot be achieved by the real system that must be 6/7 (the measured 86%) or 7/7.

4 Conclusions

In this work we propose a simple architecture for SNN self-configuration. The proposed system implements in hardware a simplified genetic algorithm. We applied the proposed architecture to temporal pattern recognition analysis. Different pseudo-random sequences were applied to different networks and the success probability was evaluated. It is shown that prediction success seems to follow a law in which the success probability is inversely proportional to the square root of the number of bits of the sequence. A new metric (the network M-index) to estimate the processing capacity of neural systems has been presented and evaluated. This metric may be used to any kind of neural system and learning process.

Acknowledgments

This work was supported by the Balearic Islands Government in part by the Regional European Development Funds (FEDER) under project PROGECIB-32A.

DOI: 10.1587/elex.5.921
Received September 06, 2008
Accepted October 17, 2008
Published November 25, 2008