Pattern recognition with Spiking Neural Networks: a simple training method

François Christophe, Tommi Mikkonen, Vafa Andalibi, Kai Koskimies, and Teemu Laukkarinen

Tampere University of Technology
Korkeakoulunkatu 1, FI-33720 Tampere, Finland
firstname.lastname@tut.fi

Abstract. As computers are getting more pervasive, software becomes transferable to different types of hardware and, at the extreme, being bio-compatible. Recent efforts in Artificial Intelligence propose that software can be trained and taught instead of "hard-coded" sequences. This paper addresses the learnability of software in the context of platforms integrating biological components. A method for training Spiking Neural Networks (SNNs) for pattern recognition is proposed, based on spike timing dependent plasticity (STDP) of connections. STDP corresponds to the way connections between neurons change according to the spiking activity in the network, and we use STDP to stimulate outputs of the network shortly after feeding it with a pattern as input, thus creating specific pathways in the network. The computational model used to test this method through simulations is developed to fit the behaviour of biological neural networks, showing the potential for training neural cells into biological processors.

Keywords: Pattern recognition, Artificial Neural Networks, Spiking Neural Networks, Computational models, Computational Biology

1 Introduction

Software is everywhere: the human environment is populated by more and more software-driven intelligent devices, connected by Internet and other networks. With the apparition of wearables and implantables, computers are getting more and more pervasive and close to the biological world. In such systems, software is expected to expand on various types of platforms.

In the current information technology, the interplay between biology and software has been indirect. Humans use software through various user interfaces, rather than with direct communication links. Concepts from biological systems have inspired various heuristic algorithms to solve computer science problems (typically optimization and search), or inspired software for communication systems to mimic the adaptive behavior of biological systems [15, 12]. Novel ways of programming by training, teaching, imitation and reward are already being

demonstrated in robotics with the help of in-silico chips behaving like neurons, i.e. neuromorphic chips [7].

The work reported in this paper is a first step in a project aiming at developing techniques to support the direct run-time interaction of biological entities and software. Our vision is that eventually software interacts directly with the biological world: software controls biological entities, and biological entities control software systems. Thus, rather than using the biological world as a model of new algorithms, we intend to let biological entities communicate directly with software. In this way, software systems and biological entities form co-operational organizations in which both parties solve problems suitable for them, contributing to a common goal. Typically, biological entities are superior in efficient massive parallel processing of fuzzy data and resilient to damage, while traditional software systems are better suited for making discrete, well-defined logical decisions. Biological entities are also far more energy-efficient than traditional computing devices.

The project focuses on integrating real neural cultures with software systems. A central problem then is the training of biological neural structures for a particular task and the connection of the trained neural culture to the software system. However, to experiment with different approaches to solve these problems, available biological neural cultures impose many practical problems: their detailed structure is difficult to study, their lifetime is limited, and they need constant nutrition. Luckily, for the past decades, Artificial Neural Networks (ANNs) have evolved to the point of being currently very close in behaviour to biological neural structures [13]. Thus, in the first stage of the project, we use ANNs to simulate biological neural networks. In a later stage we aim to transfer the techniques to biological neural cultures currently available on Multi-Electrode Arrays (MEAs) [16].

In this paper we study the basic training problem of biological neural networks using a biologically realistic model of spiking neurons. A simple pattern recognition problem is applied to this model. We demonstrate that a training technique based on Spike-Timing-Dependent-Plasticity (STDP) appears to be sufficient for these kinds of tasks.

The rest of this paper is structured as follows. In Section 2, we discuss related work. In Section 3, we introduce the computational model used in this paper. In Section 4, we evaluate the results we have obtained. In Section 5, we draw some final conclusions.

2 Related work

In [11], Maass draws a retrospective of the techniques used for modeling neural networks and presents the third generation of neural networks: SNNs. This study

classifies neural networks according to their computational units into three generation: the first generation being perceptrons based on McCulloch-Pitts neurons [4], the second generation being networks such as feedforward networks where neurons apply an "activation function", and the third generation being networks where neurons use spikes to encode information. From this retrospective, Maass presents the computational advantages of SNNs according to the computation of, first, boolean functions, and secondly according to functions with analog input and boolean output. For instance, Seung demonstrated in [14] that SNNs can be trained to behave as an XOR logical gate. The study of single neurons, the population of networks and plasticity [3] provides guidelines on how single computation units, i.e. neurons, function but more importantly on how to structure a network (e.g. number of layers, number of units in a layer) and on the models of evolution of connectivity between neurons, i.e. plasticity.

In their chapter on computing with SNNs [13], Paugam and Bohte present different methods applied for learning in SNNs. In this review chapter, they distinguish the traditional learning methods issued from previous research with ANNs, and learning methods that are emerging solely from computing with SNNs. Among the traditional methods are temporal coding, unsupervised learning such as Hebbien learning or Kohonen's self-organizing maps, and supervised learning such as error-backpropagation rules. About the "unconventional" learning methods, they are regrouped into so-called Reservoir Computing methods. Reservoir Computing methods regroup Echo State Networks and Liquid State Machines. The main characteristic of reservoir computing models relies in the apparent disorganization of the network between input and output layers. This network, the reservoir, is a recurrent network where neurons are interconnected by a random sparse set of weighted links. More over, this network is usually left untrained and only the output connections are trained and optimized according to the desired answer based on what input is given.

To a certain extent, the simple training method proposed in this paper follows the idea of training only the output layer as STDP will act alone for reinforcing the positive pathways of the network. However, the network studied in this paper is simply a feed-forward network modeled with the BRIAN simulator [2].

3 Computational model

The training method presented in this paper is studied with the computational model presented in this section because it provides a first test of feasibility before testing this method on biological Neural Networks (bioNNs). The model used for this research is composed of two basic elements: neurons and synapses. Neurons are built based on the firing model of Izhikevich which is shown to be very realistic in [9]. Synapses follow the Spike Timing Dependent Plasticity (STDP), meaning that a connection between two neurons will grow if the post-synaptic neuron fires soon after the pre-synaptic neuron. On the opposite, a connection

will decrease if the post-synaptic neuron fires before the pre-synaptic neuron. This section presents these two basic models in more details and then gives a view on the composition of the entire neural network.

3.1 Model of spiking neuron

The model of a spiking neuron used in this study is from Izhikevich [8]. This model reproduces the dynamic behavior of neurons while being computationally simple as opposed to models accounting for the structural parameters of neurons (for example, Hodgkin-Huxley model [6]). The Izhikevich model expresses the variations of electric potential in the neuron's membrane according to the current flowing through the membranes ion channels. These electric potential variations are expressed in the form of two differential equations, as follows:

$$\begin{aligned} C\frac{\mathrm{d}v}{\mathrm{d}t} &= k(v - v_r)(v - v_t) - u + I \\ \frac{\mathrm{d}u}{\mathrm{d}t} &= a(bv - u) \end{aligned} \tag{1}$$

where v represents the membrane potential, u the recovery current of the membrane, I the input current through the membrane, C the membrane capacitance, v_r the resting potential of the membrane, v_t the threshold potential for the membrane to fire a spike, k, a and b parameters adjusted according to the firing pattern required. The variables v and u of equation 1 are reset after v reaches a peak value v_{peak}, as follows:

$$\text{if } v \geq v_{peak}, \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2}$$

Figure 1 presents an example of simulation of a pyramidal neuron exhibiting regular spiking as stimulated with a constant input current of 70pA. The same simulation among others are compared with recordings from real neurons in [9] showing the precision of this models in reproducing these dynamic patterns while being computationally simple.

3.2 Model of STDP

STDP is a rule for neurons to strengthen or weaken their connections according to their degree of synchronous firing [1]. This rule mostly known in Neurobiology and Neuroscience is similar to the Hebbian learning rule widely used in learning Artificial Neural Networks and Self-Optimizing Maps [5, 10]. Considering a pre-synaptic neuron i and a post-synaptic neuron j, the STDP rule characterizes the changes in synaptic strength as:

$$\Delta w_j = \sum_{k=1}^{N} \sum_{l=1}^{N} W(t_j^l - t_i^k) \tag{3}$$

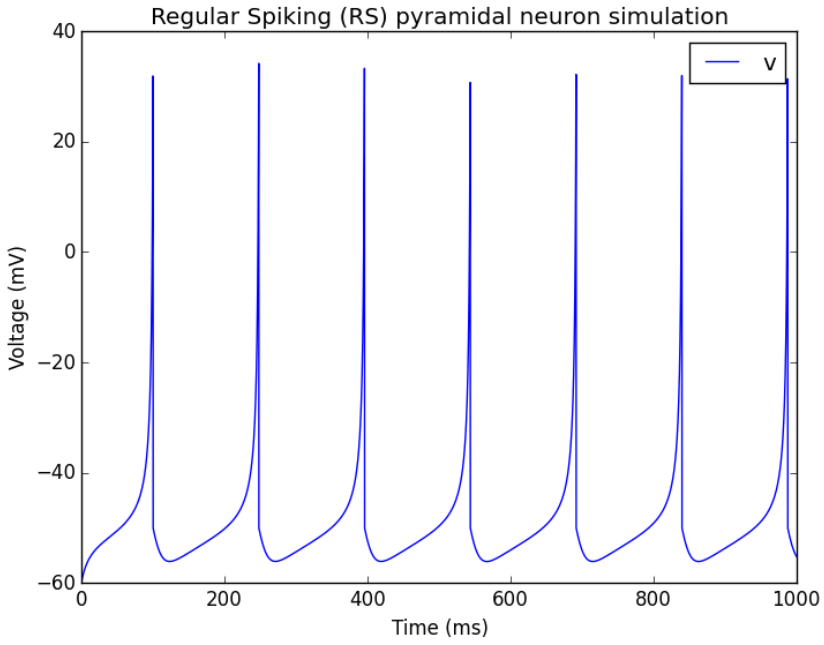


Fig. 1. Example of Regular Spiking pyramidal neuron simulated with Izhikevich model from Equations 1 and 2 (step input stimulation current $I = 70$pA from 0 to 1s). Simulated with the following values of parameters: $v_r = -60$mV, $v_t = -40$mV, $v_{peak} = 35$mV, $C = 100$pF, $k = 0.7$pA/mV^2, $a = 30$Hz, $b = -2$nS, $c = -50$mV, $d = 100$pA.

with the function $W(x)$ defining the order of decrease or increase of strength depending on the synchrony of spiking between pre- and post-synaptic neurons, expressed as:

$$W(x) = \begin{cases} A_+ \exp(-\frac{x}{\tau_+}) \text{ if } x > 0 \\ A_- \exp(\frac{x}{\tau_-}) \text{ otherwise.} \end{cases} \quad (4)$$

In equations 3 and 4, t_j^l represents the l^{th} spiking time of neuron j; similarly, t_i^k stands for the k^{th} spike timing of neuron i; A_+ and A_- are constants defining the amplitude of change in weight (at $t = 0_+$ and $t = 0_-$, respectively); and, τ_+ and τ_- are time constants of the exponential decrease in weight change.

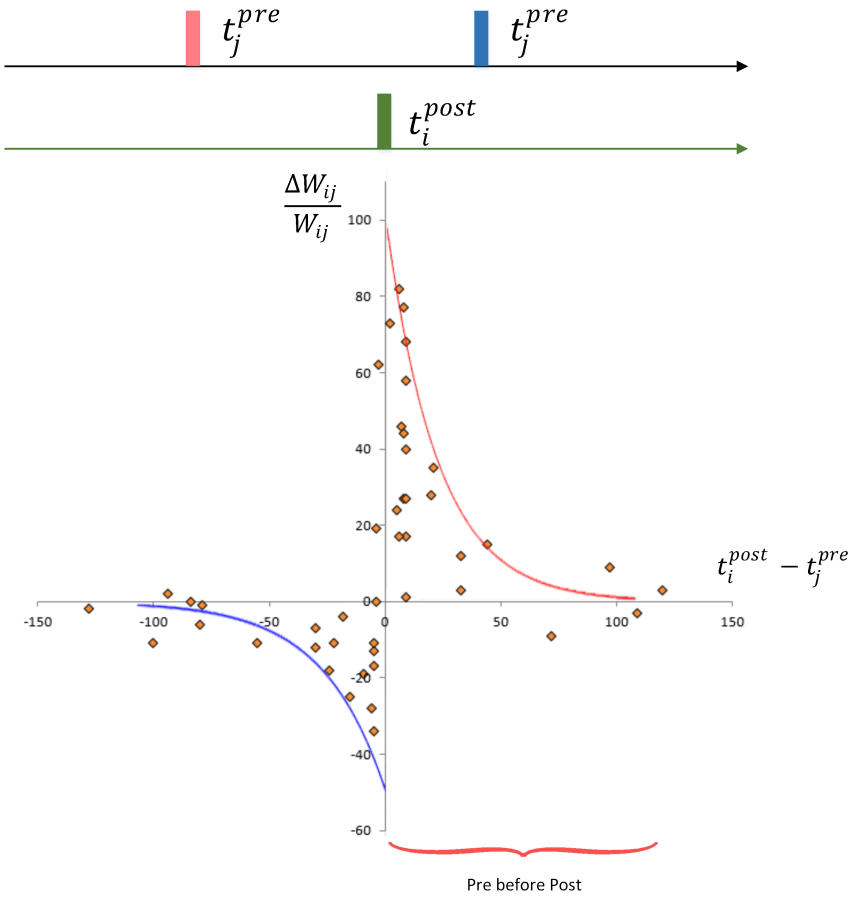


Fig. 2. Model of STDP (reproduced after Bi and Poo [1])

Figure 2 represents equation 4 of relative weight changes in relation with the experiments realized by Bi and Poo in [1]. This figure shows typically a decrease in synaptic weight when the pre-synaptic neuron spikes after the post-synaptic neuron, and on the opposite an increase in connectivity from pre- to post-synaptic neuron if the pre-synaptic neuron spikes just before the post-synaptic neuron.

3.3 Network model

The network developed in this study as an example of pattern recognition is presented in Fig. 3. This network is dedicated at recognizing patterns from a 5x5 pixels image. It is composed of:

- 25 input neurons corresponding to each pixel of the image,
- a hidden layer of 5 neurons, and
- 2 output neurons (1 corresponding to the neuron reacting when a circle appears in the image, the other one being a test neuron for comparison between learning and no stimulation).

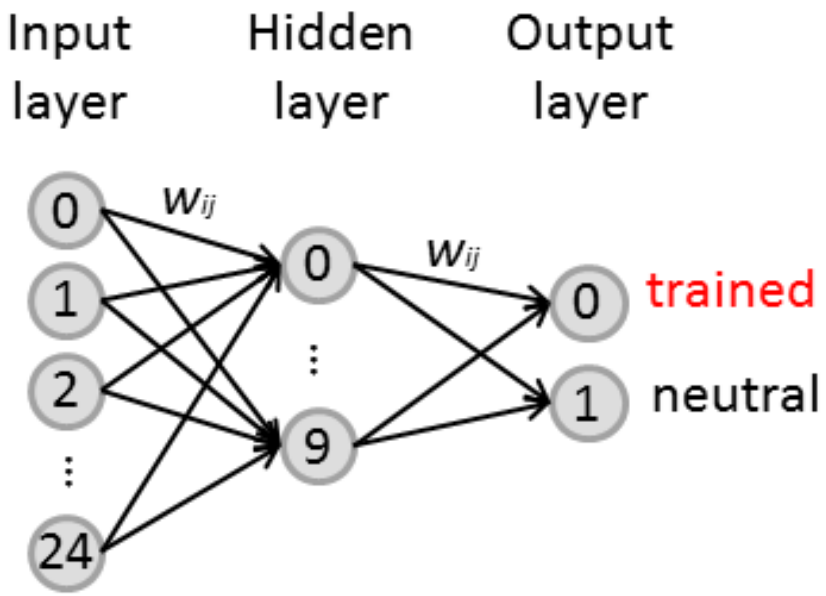


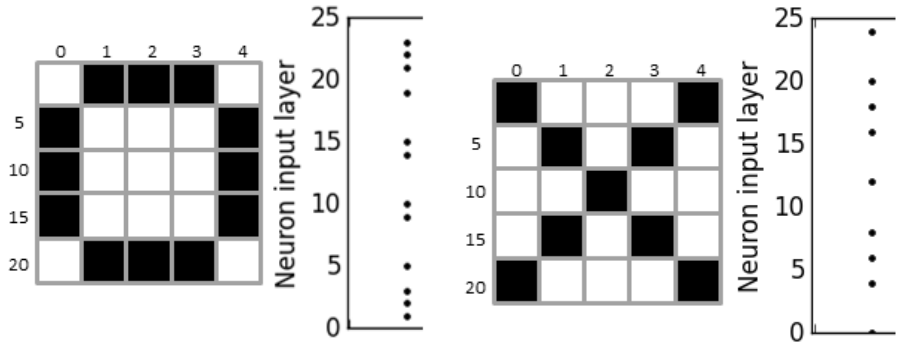
Fig. 3. Representation of the network developed for 1 pattern recognition

4 Training method and Evaluation

This section goes more into the details of the simple learning method using STDP rule for training the network. A pattern recognition task is used as a case study for testing the learning method proposed.

4.1 Pattern recognition task

The pattern recognition task evaluated during this simple case study consists in making the difference between a circle given as input stimuli and other inputs (in that case an X-cross shape). These shapes are represented as coming from a 25 pixels image (5x5 matrix), each pixel being binary: black or white. Figure 4 presents the input circle and X-cross patterns and their respective representations as input stimuli to the network.



(a) Circle pattern (b) X-cross pattern

Fig. 4. Input patterns as images and their respective representations as input stimuli

4.2 Training method and test protocol

After initial tests on the learning ability of the network, the training period was adjusted to be 15s during which the input layer is stimulated every 100ms with a circle pattern. 10ms after stimulating the input layer, the output neuron is given an external stimulation making it to spike. This spiking, in relation with the preliminary spiking of neurons from the input layer, reinforces the paths between activated neurons of the input layer and the trained neuron of the output layer. This training can be seen in the first phase of the time diagrams (top and middle) of Figure 5 from $t = 0$ to 15s.

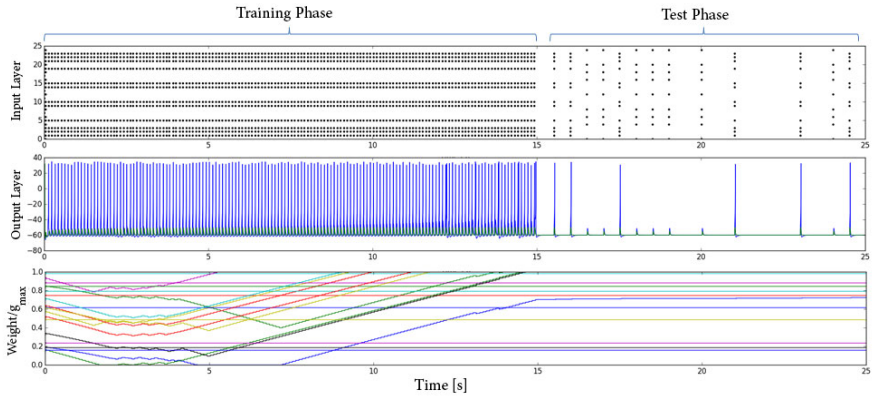


Fig. 5. Training phase and test phase of the circle recognition experiment

The testing phase is composed of 6 stimuli with circle pattern and 7 stimuli with a different pattern (in that case representing an X-cross). These stimuli of the input layer happen without any external stimulation of the output layer. The neuron trained for recognizing a circle fires on its own after the learning phase. These test patterns are sent between $t = 15,5$ to 25s with the following sequence: {circle, circle, cross, cross, circle, cross, cross, cross, cross, circle, circle, cross, circle} at the respective times {15.5, 16, 16.5, 17, 17.5, 18, 18.5, 19, 20, 21, 23, 24, 24.5} seconds. The two upper time diagrams of Figure 5 show this test phase.

The third time diagram of Figure 5 (down) presents the evolution of strength of synapses between neurons from the hidden layer and neurons from the output layer. This evolution shows first a stabilization period from the random strengths given as initial condition to lower values. Secondly, learning can be seen as the strengths of certain synapses increase to high levels of connectivity (i.e. to levels higher than 0.8 times the maximum connectivity and often reaching this maximum).

4.3 Results

The 1000 simulations of this experiment revealed a success rate of 80.3% in recognizing a circle from a cross. This rate was computed after the execution of a thousand experiments. From these experiments, five different cases were observed:

- correct learning: output fires only when a circle is given as input (80.3%)
- some mistakes: output fires sometimes when input is a cross (5.7%)
- always firing: output always fires whatever the input may be (12.8%)
- no learning: output never fires (1.1%)

– wrong learning: output fires only when input is a cross (0.1%)

These different cases are represented in Figure 6.

These results show a high learning rate, i.e. a high rate of correct experiments (80.3%), meaning that such learning method has correct grounds. Indeed, there is space for improving this rate and a lot to learn from the analysis of failed experiments.

First, we can notice that the network keeps on learning even after the training phase has stopped. Each time a pattern is fed as input to the network, small increase in synaptic weights take place.

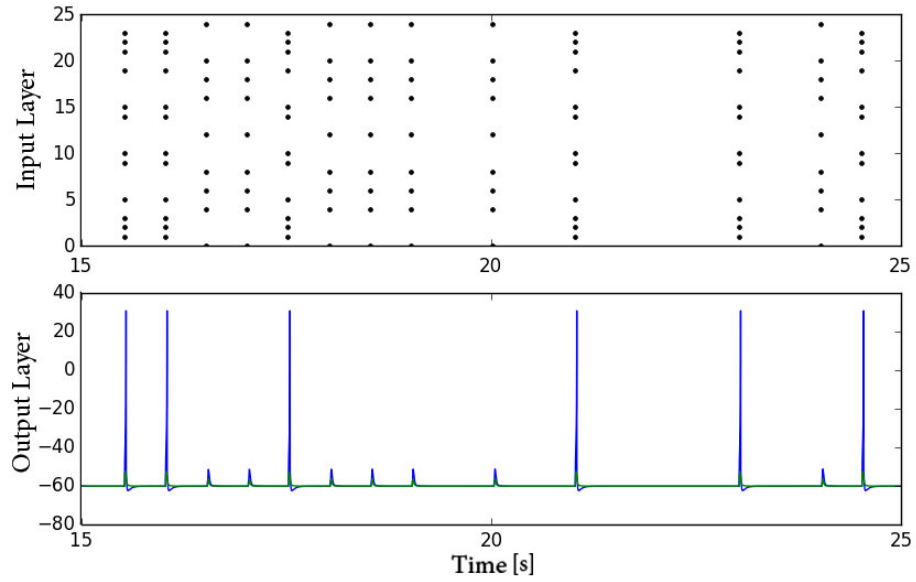
The second important thing noticed due to this continuous learning is that the output neuron trained to recognize a circle also gets trained when receiving another pattern as input. As the synaptic levels may already be high, it requires only few X-cross stimulation signals for the output neuron to start spiking and we notice that when it has learned to spike for a cross pattern, it will then fire each time a cross appears. This is what happens for the cases where some mistakes are found (57 cases out of 1000 simulations).

Third, for 128 simulations the synaptic levels are high from the beginning of the training due to the initial random value of synaptic weights. This causes the network to be "over-trained" and thus to fire for every kind of patterns from the beginning of the test phase. In the contrary, for 11 simulations the output neuron does not fire at all when given any input. These 11 tests show clearly low rates in synaptic weights and low increase in synaptic weights during training expressing the fact that the network does not have time to learn during this period of time. The only simulation where the output neuron fires only and always when given the wrong input could not be explained.

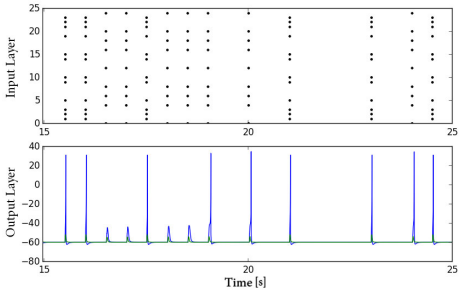
5 Discussion and future work

We have studied a central problem in using biological neural networks as computing resources: how to train the neural culture for a particular task. The SNN presented in this paper shows reasonable success rate in learning (80.3%) to differentiate between two patterns. As the behavior of the simulated network is very close to that of real biological neural networks (bioNNs), this experiment gives preliminary insight for using bioNNs to process complex tasks requiring massive parallelism.

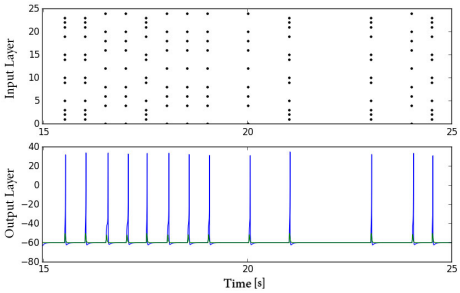
However, there is still possibility for improvement in performing such recognition tasks. In the case where pathways did not have time to form during initial training, it is indeed possible to continue training the network until synaptic level reaches the appropriate levels. On the opposite case, when the network fires for any type of input pattern, meaning that it is "over-trained", training the network with negative output stimulation should help restoring the differentiation



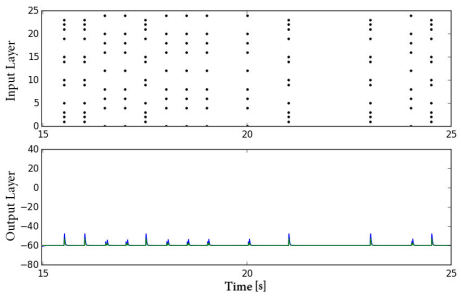
(a) Correct learning (rate 80.3%)



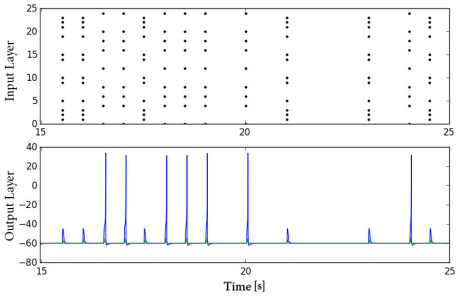
(b) Some mistakes (rate 5.7%)



(c) Always firing (rate 12.8%)



(d) No learning (rate 1.1%)



(e) Wrong learning (rate 0.1%)

Fig. 6. Example of the 5 different cases found in experiments

between input patterns. Such negative stimulation can be realized by stimulating the output neuron just prior to the input pattern, when an input pattern is to be discarded by the network. This way, the synaptic pathways related to this input pattern would decrease due to the STDP rule, thus de-correlating the output neuron with this input pattern.

The proximity in behavior of SNNs from bioNNs should not require efforts to transfer computations from silicon platforms into biological platforms. From this point, transferring the various tasks developed for the past sixty years with ANNs (e.g. classifiers, generators, etc.) to small bio-computers will be possible. Research directions for such transfer lead to the following questions:

- Can a taxonomy of the various tasks performed with ANNs and their hierarchical relations be developed?
- Can we classify tasks as unitary/atomic to some higher level? On the other hand, can tasks be broken down into summation of unitary tasks?
- Is it possible to automatically associate tasks to network structures (number of nodes, number of layers etc.) ?
- Can training also be automatically generated for these specific tasks?

These questions could lead to the construction of a compiler deriving the number of biological neural networks used for an application, their structural parameters and the training associated to each task required for the application.

6 Conclusion

In this paper, we presented a simple learning method using STDP for training pathways of a SNN. This method was tested on a SNN model trained to differentiate between two patterns given as input to the network. The results of this test (80.3% success rate) combined with the fact that the behaviour of the simulated network is very close to the one of a real biological network gives promising expectations for the future of this project. This first test is still a preliminary test towards applying bioNNs to the computation of more complex tasks such as handwritten digit and character recognition [1]. Expectations on the results to such test should give similar success rate as to the test conducted in this paper. Next experiments will be conducted on real biological cells in order to validate the possibility of training bioNNs for pattern recognition tasks. In the final stage we intend to connect such trained bioNNs with software applications requiring pattern recognition capability such as classification of moving objects.

Acknowledgement

This research is funded by the Academy of Finland under project named “Bio-integrated Software Development for Adaptive Sensor Networks”, project number 278882.

[1] Such training is usually tested on the MNIST dataset available from http://yann.lecun.com/exdb/mnist/

References

1. Bi, G.Q., Poo, M.M.: Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. The Journal of neuroscience : the official journal of the Society for Neuroscience 18, 10464–10472 (1998)
2. Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M., Morrison, A., Goodman, P.H., Harris, F.C., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A.P., El Boustani, S., Destexhe, A.: Simulation of networks of spiking neurons: A review of tools and strategies (2007)
3. Gerstner, W., Kistler, W.: Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press (2002)
4. Hayman, S.: The mcculloch-pitts model. IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339) 6 (1999)
5. Hebb, D.O.: The Organization of Behaviour: A neuropsychological theory. Wiley (1949)
6. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. Journal of Physiology pp. 500–544 (1952)
7. Indiveri, G., Linares-Barranco, B., Hamilton, T.J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., Boahen, K.: Neuromorphic silicon neuron circuits. Frontiers in neuroscience 5(May), 73 (1 2011)
8. Izhikevich, E.M.: Simple model of spiking neurons. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 14(6), 1569–72 (1 2003)
9. Izhikevich, E.M.: Dynamical Systems in Neuroscience : The Geometry of Excitability and Bursting, Chapter 8. The MIT Press (2007)
10. Kohonen, T.: The self-organizing map 21(May), 1–6 (1998)
11. Maass, W.: Networks of spiking neurons: The third generation of neural network models. Neural Networks 10(9), 1659–1671 (1997)
12. Nakano, T., Suda, T.: Self-organizing network services 16(5), 1269–1278 (2005)
13. Paugam-Moisy, H., Bohte, S.: Computing with spiking neuron networks. In: Handbook of Natural Computing, pp. 335–376 (2012)
14. Seung, H.S.: Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. Neuron 40(6), 1063–1073 (2003)
15. Suzuki, J., Suda, T.: A middleware platform for a biologically inspired network architecture supporting autonomous 23(2), 249–260 (2005)
16. Taketani, M., Baudry, M. (eds.): Advances in Network Electrophysiology: Using Multi-electrode Arrays. Springer (2006)