Representing and Decoding Rank Order Codes Using Polychronization in a Network of Spiking Neurons

Francesco Galluppi and Steve Furber

Abstract— **The introduction of axonal delays in networks of spiking neurons has enhanced the representational capabilities of neural networks, whilst also providing more biological realism. Approaches in neural coding such as rank order coding and polychronization have exploited the precise timing of action potential observed in real neurons. In a rank order code information is coded in the order of firing of a pool of neurons; on the other hand with polychronization it is the time of arrival of different spikes at the postsynaptic neuron which triggers different post-synaptic responses, with the axonal delays compensating for different timings in the afferents. In this paper we propose a model in which rank order coding is used to represent an arbitrary symbol, and a polychronous layer is used to decode, represent and recall that symbol. To prove that the polychronous layer is able to do this a detector neuron is trained with a supervised learning strategy and associated with a single code. According to this premise the detector neuron only fires on the appearance of the associated code, even in the presence of noise. Tests prove that rank order coding and polychronization can be coupled to code and decode information such as intensity or significance using timing information in spiking neural networks in an effective way.**

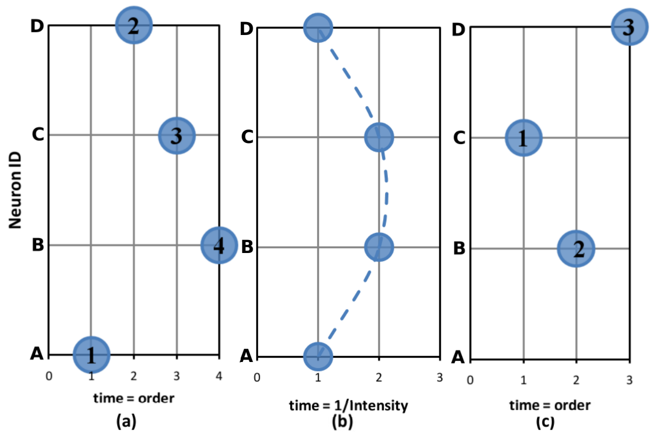


Fig. 1. Time coding, circles represent neurons firing, numbers inside represent order of firing: (a) rank order code, neuron firing times are ordered by significance with no repetition [A, D, C, B] (b) rank order code where the firing time is proportional to the intensity of the stimulation (in this case couples A-D and B-C code for two different values of intensity) (c) N-of-M (3-of-4) rank order code, 3 neurons out of the 4 fire in the order C-B-D

Introduction

THE third generation of neural networks has introduced a novel set of modelling functions and parameters which both model biological neurons more precisely [12] and increase the computational power of networks of neurons [17]. Millisecond precise fire timing has been observed in groups of neurons recorded *in vivo* [21]. Precise time coded pulses and the introduction of axonal delays have been proved to enhance the representational power of networks of spiking neurons [19]. Consequently models taking advantage of the timing distribution of spikes have been produced. Some of them, such as rank order coding [31], exploit the possibility to code information in the timings of the spikes in order to take account of the fast recognition responses of the visual system [30]. At the receiving end polychronization [13] has introduced as a framework to explore how convergent delays and plasticity can generate time-locked group patterns, and has proven to have a great representational power [15]. Polychronization is based on the concept that postsynaptic neurons are sensitive to precise timings of their afferents due to axonal delays, which can compensate for latencies in the input neurons and enhance postsynaptic response.

In the view presented in this work the two models are integrated in a complementary system, rank order coding being used by an input population to code information and polychronization being used by a higher dimensional population of postsynaptic polychronous neurons. This population is able to represent the codes presented in the presynaptic input layer, and a neuron can be trained to discriminate an arbitrary selected code.

The rest of the paper is structured as follows: rank order coding and polychronization are presented in sections I and II. A way to couple them in order to code/represent/decode information is presented in section III, while the model architecture is presented in section IV. A description of the experiments conducted and details about the implementation are given in sections V and VI respectively. Considerations of the model and how to further investigate it are presented in the Discussion section. Finally our conclusions are made in the last section.

The authors are with the School of Computer Science, The University of Manchester, Manchester, UK (email: {francesco.galluppi}@cs.man.ac.uk)

I. Rank Order Coding

Rank order coding has been introduced to take account of the processing speed of the feed-forward recognition networks involved in visual perception [31]. The input is coded in the order of firing of the input population. Rank order coding has proven to be a biologically plausible explanation in several tasks involving image recognition [7] and reconstruction [25] by coding information in the spike order [35]. First spikes have also been considered to carry more information, confirming the possibility of fast recognition and restoration with few spikes [24]. Rank order coding shares properties with unordered *N*-of-*M* codes and have been coded in a sparse distributed memory [9].

Examples of how to code information in the timings of spikes are given in figure 1. A rank order code can be expressed as a sequence (with no repetition) of the input neurons (case *a*, neurons firing in the order A-D-C-B). If two neurons can have the same position the rank order code can be thought as coding the intensity of the stimulation: neurons that receive a more intense stimulation fire first [35] (case *b*, neurons A and D receiving more stimulation than neurons B and C). If only a subset of the input population is considered information can be code in an *N*-of-*M* rank order code [9] (case *c*, 3-of-4 code with neurons firing in the order C-B-D).

Shunt inhibition [3] is often used to the decode a rank order code: by decreasing the effectiveness of a synapse with the number of spikes received, discrimination can be achieved [8]. In this paper we use polychronization as an alternative neural structure which is able to represent and decode the order of spikes from a source population.

II. Polychronization

Polychronization has been introduced to explain the memory capacity of spiking neural networks with delays and plasticity [13]. Information is stored as polychronous groups, time-locked firing patterns involving a group of neurons [15]. While rank order coding can be used to exploit the temporal properties in the source (presynaptic) neurons, polychronization can be used to organize and tune postsynaptic neurons to compensate for time differences in the inputs using axonal delays. Converging spikes will increase the neuron response. Polychronous neurons will then be sensitive to certain time-locked input patterns, but not to others.

The idea of polychronization is illustrated in its simplest form in figure 2. The neurons are interconnected with different delays so that only precise firing time patterns of the input neurons will have their inputs converge onto the output neurons at the same moment. For instance in case *d* if input neurons fire at 2, 3 and 4 ms respectively their spikes will converge onto neuron B at the same time and arrive in sparse order to neuron A.

III. Coupling rank order coding and polychronization

In the previous two sections we have seen how rank order coding can be used to encode information in the spike timings of an input population and how polychronization can be used by a postsynaptic neuron to decode a temporal pattern emitted by a source population by compensating for the latencies with the axonal delays.

Polychronization then seems a natural candidate to decode rank order coded inputs. If we look again at figure 2 we can think of the firing patterns in neurons 1, 2 and 3 as two distinct rank order codes, and the two output neurons A and B as responders to a particular rank order code.

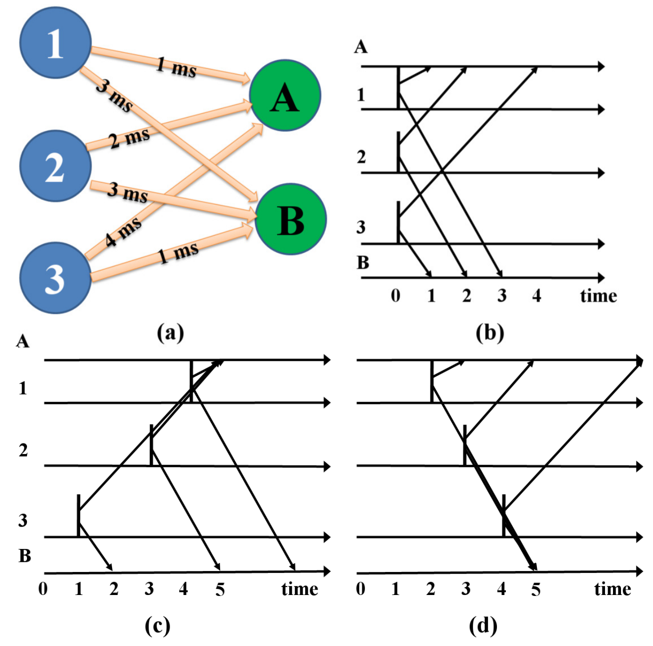


Fig. 2. (a) Example network: neuron A receives input from (blue) neurons 1, 2 and 3 with delays 1, 2 and 4 msec; neuron B receives inputs with delays 3, 2 and 1 msec respectively (b) if input neurons fire together their contribution arrives in sparse order to the postsynaptic neurons (green) eliciting a weak response (c) if neurons fire in a precise time pattern (at msec 4, 3 and 1 respectively) their spikes converge to neuron A due to compensation through the axonal delay while having no effect on neuron B (d) if input neurons fire at 2, 3 and 4 msec their spikes will converge onto neuron B and arrive in sparse order to neuron A

Such a network can be devised to decode rank-order codes with a priori assignment of firing semantics to each polychronous neuron by tuning the delays manually. Instead in this work a polychronous layer is connected to the input source with random delays so as to act as a spiking neural implementation of a sparse distributed memory [9] which is able to store and recall rank order codes. In this sense the system can be considered like a spiking implementation of liquid state machine where the polychronous layer represents the liquid and the trained output neuron the readout function [20].

IV. Proposed Model Architecture

In order to test the hypothesis that rank order coding and polychronization can be coupled to effectively represent information in a neural network a 3-layer model is proposed. All neurons are modelled as Leaky Integrate-and-Fire (LIF).

The first layer is a population composed of 10 neurons. The coding strategies will determine the firings in the input layer. We have tested the system with a *7*-of-*10* rank order code (like the one shown in 1*c* with N=7 and M=10) and then with rank order code coding for intensity. In this case all the input neurons in the population layer fired according to the intensity of the stimulation (example b in figure 1).

We then divide the process in two phases: in the first one (*tuning*) Spike-Timing Dependent Plasticity (STDP) [27] [2] is used to tune the weights between the input and the

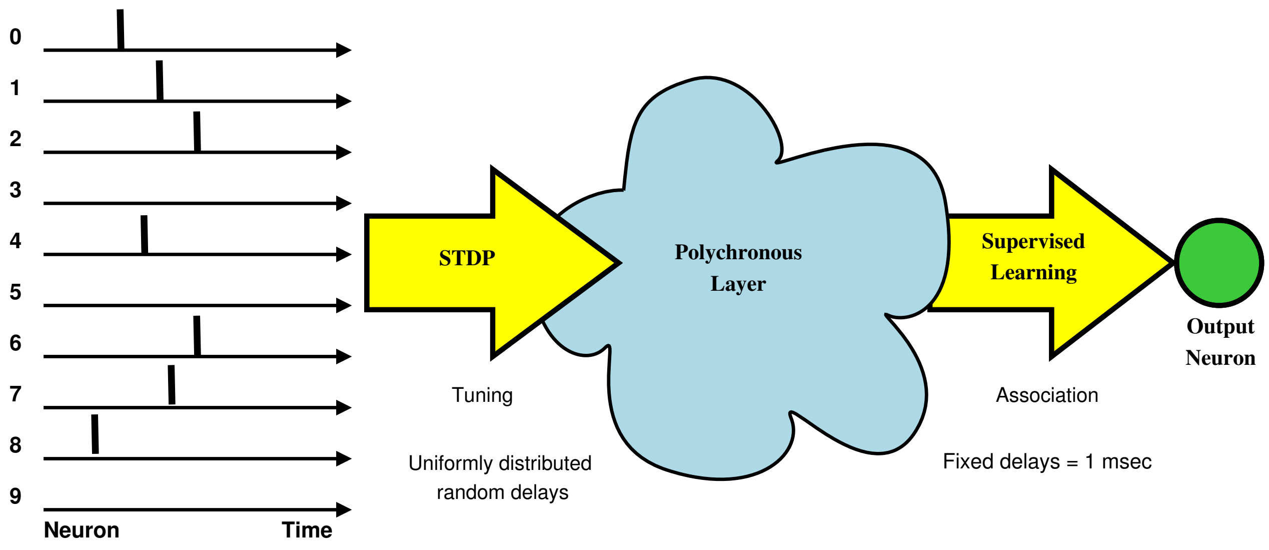


Fig. 3. Model architecture: in the tuning phase 200 random codes are presented at the input layer while STDP is turned on. Neurons in the polychronous layer respond to different time-locked properties of the input (parts of the input code with convergent delays will make a polychronous neuron fire). In the association phase the combination of the active polychronous neurons corresponding to the selected code are learned by the output neuron through a supervised learning process

polychronous layer, while in the second one (*association*) an output neuron is associated with an arbitrary input code through a supervised learning strategy.

The first layer connects through an STDP projection to the polychronous layer, which is formed by 1000 LIF neurons, intrinsically connected by inhibitory connections in a Winner-Take-All [18] pattern in order to avoid the possibility that two neurons should learn the same pattern. The connection is made with 30% probability, and the delays are uniformly distributed in the range of the length of the code, letting the postsynaptic neuron potentiate subportions of the codes with convergent delays. In this way different neurons in the polychronous layer will detect different parts of the input code, according to how delays converge on them.

Each code will then be represented by a different set of neurons in the polychronous layer responding to a precise time sequence of the inputs. In this sense the polychronous layer acts as a sparse distributed memory [16]: the input is represented in a higher dimensional space where the elements respond to parts of the input code, according to how much their delays match part of the input pattern. These connections are trained with STDP in a preliminary tuning phase where 200 randomly picked symbols are submitted to the network, in order to make the neurons respond adaptively to different convergent delays. After this phase STDP is turned off, and the polychronous layer is capable of representing and discriminating different input codes.

To prove our hypothesis we implemented a supervised learning strategy the only purpose of which is to demonstrate that the polychronous layer is capable to represent a rank order code. To verify this an output neuron can be associated with an arbitrary code which can discriminate it against other codes and noise. This supervised learning phase is used to

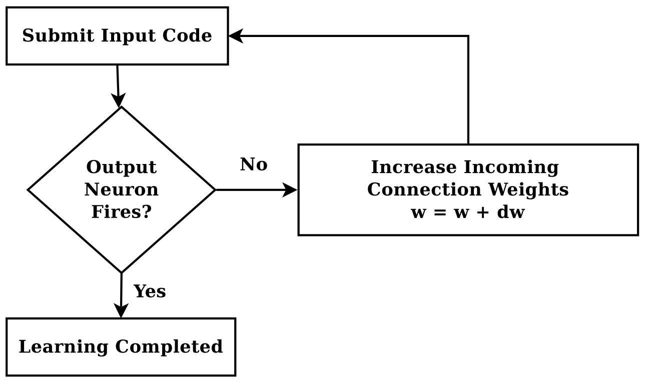


Fig. 4. Supervised Learning Algorithm: weights are initially set to 0. The input code to be associated with the output neuron is presented. If the output neuron fires learning is completed. Otherwise all the incoming connections from the polychronous layer contributing to the input code are potentiated and the input is presented again. The process is repeated until the output neuron fires.

map the output neuron to a combination of the activity in the polychronous layer so as to make the output (detector) neuron respond only to a particular combination of afferents. The learning is based on changing the weights according to an external supervised signal. Therefore it does not exploit the timing code of the system (it is independent from time) and it doesn't affect the timing behaviour of the model (all delays are set to 1 msec). In this way is possible to extract the information stored in the polychronous sparse distributed memory layer.

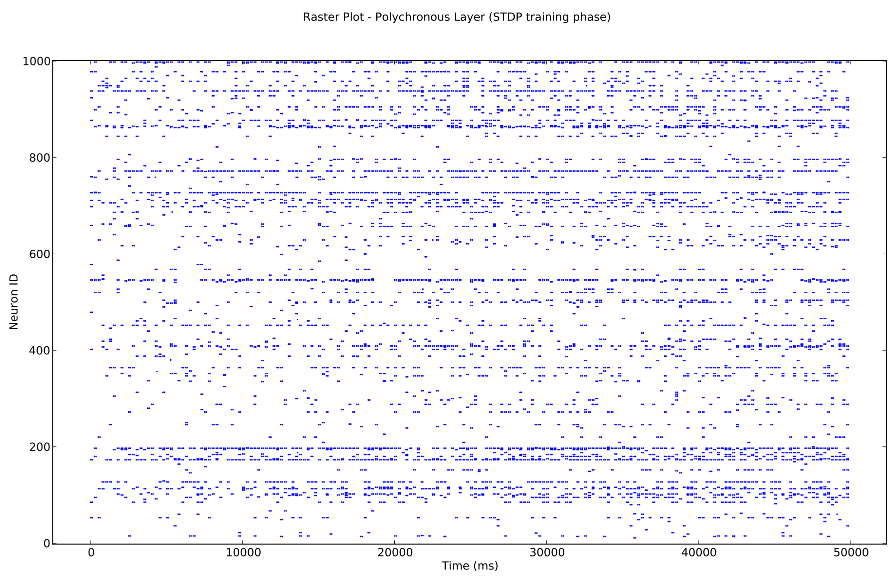


Fig. 5. STDP tuning phase in the polychronous layer: the combination of neurons activated in the polychronous layer is different for every code. 200 codes are used in the tuning phase, every code lasts 250 msec

V. Simulations and Results

A. Tuning the polychronous layer with STDP

Simulations were carried as follows: at the beginning the network is trained with 200 random symbols coded as described in the previous section in order to let STDP shape the connections. An example of this tuning phase is given in figure 5 where it can be noted that, although some input codes share some neurons, the combination of neurons activated in the polychronous layer is different for every code. As a net effect STDP potentiated 30% of the connections. After this phase STDP is turned off, and the only learning that occurs is in the next supervised learning association phase.

B. Associating a code to a detector neuron with supervised learning

In this phase a code is selected to be learned and is presented at the input layer repeatedly, while a supervised learning algorithm is used to map the desired combination of input to the output neuron [23]. The weights are changed upon the receipt of an external control signal to train the output neuron to respond to the combination of neurons firing in the polychronous layer (the weights in the polychronous layer don't change since STDP is turned off). All the neurons from the polychronous layer project to the output neuron with an initial weight set to 0 in order to let the algorithm potentiate only the ones that contribute to the right code. The supervised learning algorithm used is structured as follows (cfr. fig 4):

- the code is presented at the input layer and activates a subset of neurons in the polychronous layer
- if the output neuron fires learning is complete
- else the weights from the active polychronous neurons and the output detector neuron are increased and the process is repeated

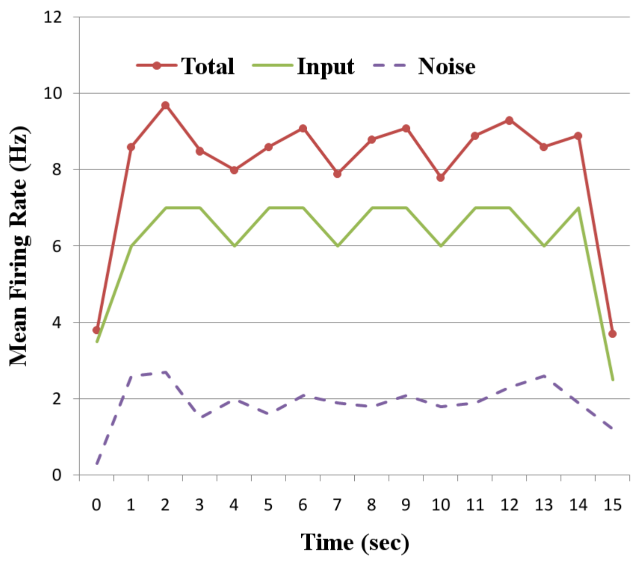


Fig. 6. Contribution in terms of population mean firing rate of noise to the normal activity of the input population

Such a process ensures that the output neuron will fire only when the combination of neurons in the polychronous layer is detected, hence mapping the combination of input to the output. It also does not alter the timing properties of the system since all the delays are set to 1 msec. An example of the supervised learning phase is presented in figure 7.

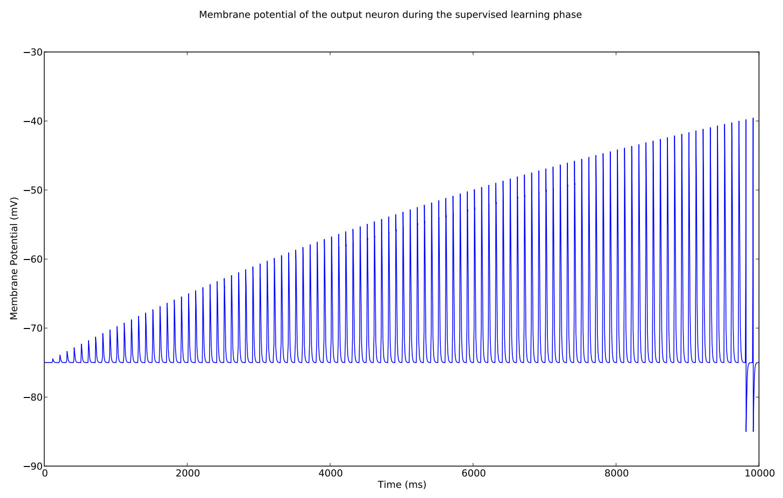
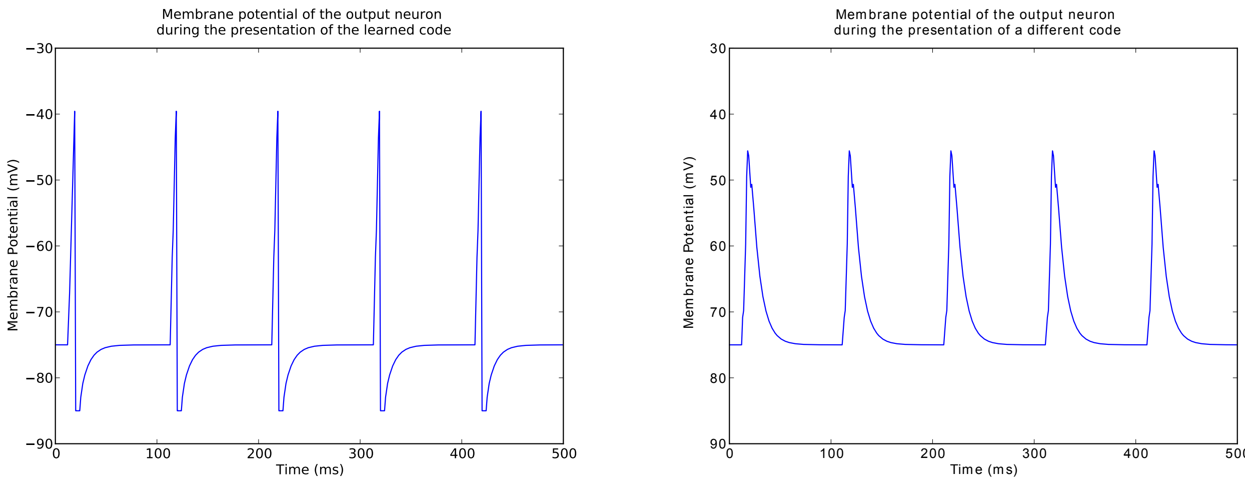


Fig. 7. The output neuron is trained to respond only when the combination of firings in the polychronous layer appears. This is done by increasing the weights from the active polychronous neurons to the detector neuron until a spike is produced (in this case at sec 10 of the simulation)



(a) The output neuron fires when the learned pattern is presented

(b) A code obtained by the permutation of 2 positions in the learned code is not able to cause the output neuron to cross the threshold potential (-40 mV)

Fig. 8. Membrane potential of the output neuron during the presentation of two codes

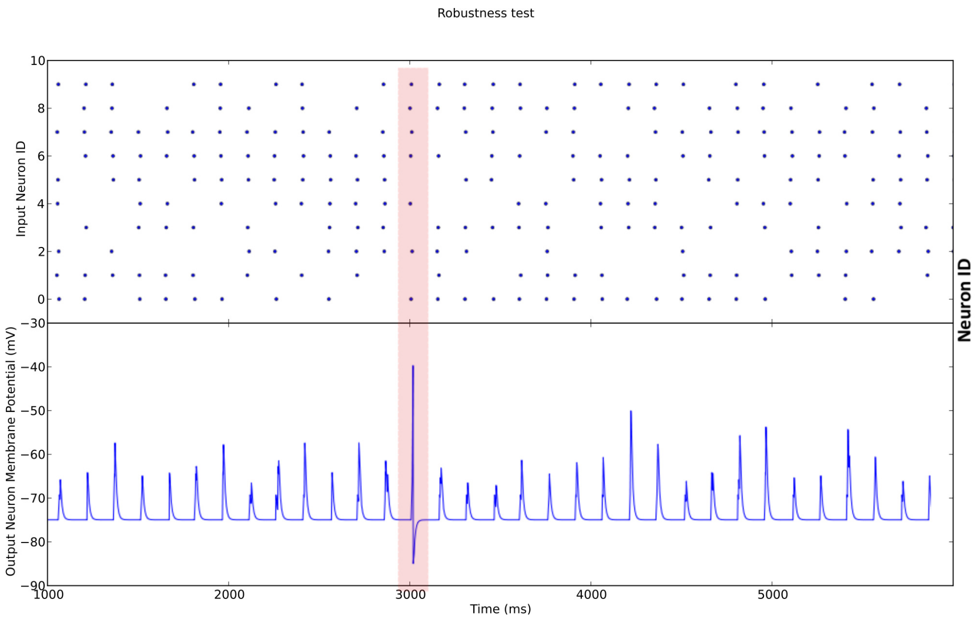
C. *Testing N-of-M rank order codes*

In this test we trained the output neuron to respond to a rank order code of 7 out of 10 neurons firing each with a different position encoded in the firing time (see fig 1(c) and fig 9). As predicted, after the output neuron is trained it only responds to the combination of firing neurons in the polychronous layer associated with the input code, otherwise it exhibits sub-threshold oscillations but stays silent. An example of the output neuron membrane potential in the contrasting case of the presentation of the matching code and of a code with two positions interchanged is presented in figure 8(a) and 8(b) respectively.

Since the process can be repeated with as many output neurons and input codes as desired, is possible to train an arbitrary number of output neurons to respond to different input codes. To test the robustness of the learning we tested the network with 1000 different inputs, inserting the learned input code amongst them. Results of the test are reported in figure 9. It can be seen that only when the learned code (on the right) is presented the output neurons fire; all the other codes are not capable of causing the output neuron to cross its firing threshold, set at -40 mV.

D. *Testing intensity rank order codes in a noisy regime*

We repeated the procedure for a network coding the intensity of the stimulus at the input layer by using rank order coding [35] as shown in fig1(b) and fig 6. The learned pattern can be observed in figure 10. After the learning phase we observed that the output neuron was responding only to the selected code, as in the previous experiments, so we added a



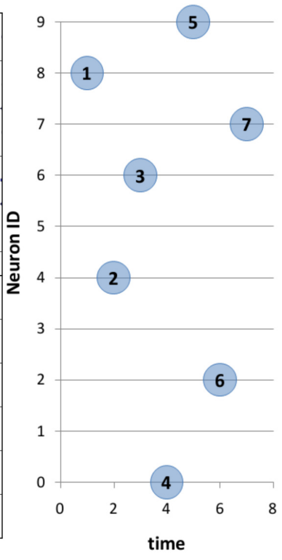


Fig. 9. The membrane potential of the output neuron crosses the threshold only when the learned code is presented (red box). All the other codes don't elicit any spike in the output neuron (only the first 10 sec of the simulation are shown 60/1000 tested patterns). On the right the representation of the learned pattern 8-4-6-0-9-2-7 (order of firings)

Poisson noise source at the input layer with mean firing rate of 2Hz (equal to 25% of the input mean fire rate, as shown in fig 6). Despite of the noise the output neuron is still able to respond only to the learned code, as shown in figure 10. Increasing the noise above 30% of the original signal led to false detection (results not shown - 2 false detections on 100 symbols tested).

VI. Materials and Methods

A. Common features

All simulations have were carried on the NEURON simulator [1] [11], using pyNN [5] as a front end so as to be able to migrate easily the model onto other simulators or to translate and run it on neuromorphic hardware [4] [10].

All simulations share this set of features: we use the Leaky Integrate-and-Fire (LIF) neuron equipped with current-based first-order kinetic synapses [6], corresponding to the `IF_curr_exp` standard neuron model in pyNN. The input population is composed of ten neurons which fire according to the coding scheme selected. The polychronous layer is composed of 1000 neurons. Connections between the input layer and the polychronous layer are made with 30% probability; the polychronous population neurons have also been interconnected with inhibitory connections ($weight = -2$). The output neuron is also a LIF neuron, sharing all parameters with the rest of the neurons except for the reset voltage and the refractory period. The STDP uses the `SpikePairRule` model with the following parameters $A_+ = A_- = 0.01, \tau_+ = \tau_- = 100$ msec. Stimuli are sent every 100 msec, and the STDP time window is set accordingly. During the STDP tuning phase weights cannot be potentiated more than 25% of their initial weight. The change in weight during the association phase is set to $\Delta w = 0.001$ and the initial weight is set to 0. The simulation time step is calculated with millisecond precision.

B. N-of-M rank order codes

All the cells in the model share the following parameters (written in pyNN format): $tau\_m : 8, v\_init : -75, v\_rest : -75, v\_thresh : -45, tau\_syn\_E : 1, tau\_syn\_I : 1, cm : 1.5$. Input is organized as follows: 7 out of 10 neurons are picked in a random order and they fire every 2 msec accordingly to their position in the rank order code (see fig 9). Delays between the input and the polychronous layer are set in the range 1-14 msec while the weight is set to 14.

C. Rank order coding intensity in a noisy regime

Neuron parameters are the same as the previous section except for the membrane threshold which is set at -40 mV. Input is organized as follows: intensity can be coded in 4 different values, which will be inversely proportional to the firing time so that the neurons coding for the highest intensity will fire first. Four different levels of intensity are encoded with a time step of 2 msec (see fig 10). To cover the whole pattern delays are uniformly set in the range 1-8 msec while the initial weights are set to 9. Noise has been modelled as a Poisson process with $rate = 2Hz$ by a `SpikeSourcePoisson` population injecting into the input population.

VII. Discussion

The results of the tests show that rank order coding and polychronization can be used together as an efficient code representation strategy. The supervised learning algorithm proposed in the model shows that the information is encoded

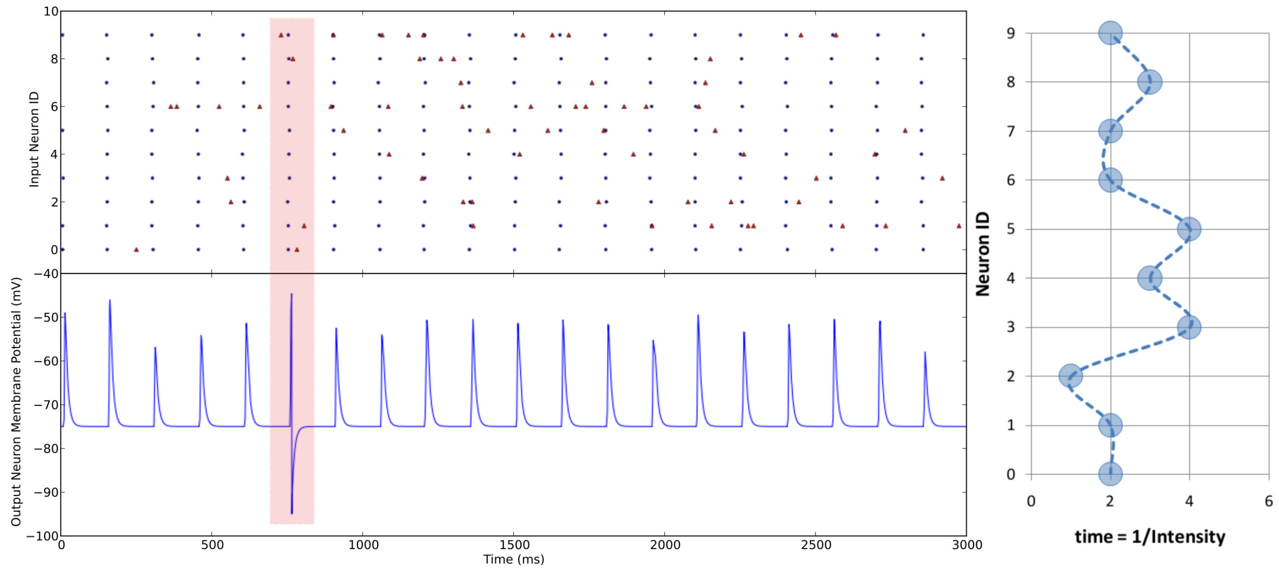


Fig. 10. The output neuron fires only when the learned code is presented (red box). All the other codes don't elicit any spike in the output neuron, even in the presence of the noise (red triangles in the input raster plot). The pattern tested is presented in the right part of the figure as coding intensity of the stimulation (1/intensity so that neurons that receive more intense input fire first, in this case is neuron 2)

only in the polychronous layer. It is different from previous proposed models [32] [7] because it uses a polychronous layer as a spiking neural implementation of a sparse distributed memory [9] to represent the code and a supervised learning strategy to learn it, rather than using shunt inhibition [3]. Even in this basic form the learning has confirmed that the polychronous layer is capable of representing rank order codes in a way that can be disambiguated by a detector neuron.

More realistic methods of association can be implemented, for instance by using a STDP regulated by a signal such as dopamine [14]. Such a control signal can be externally generated in supervised learning algorithm or can be generated by another portion of the network such as a value system [28] in an unsupervised adaptive learning paradigm.

The probability connection proved to be a very important parameter. Lowering the connection number between the input and the polychronous layer proved to lead to less discrimination power in the system. Variation in all the time parameters of the system (time constants, delays) led to different behaviours. A more formal definition of the system could be researched, in order to optimize the parameters of the model.

The polychronous layer exploits the capabilities of polychronization to a minimum extent, in the sense that there is no plastic internal connection that can lead to the formation of polychronous groups inside the layer itself [15]. Further tests in this direction are needed to verify the possibility of associating different code sequences represented by polychronous groups within the polychronous layer.

If the spikes are considered to be produced by the same source the polychronous layer can code for different patterns where fine recognition is given by integrating a source with different timings such that time codes significance. But the same input can be considered to be generated by different source areas each coding a different feature in parallel in functionally specialized maps [34]. The polychronous layer could be a converging neural association area such as the pre-frontal cortex [22]. Integration from different areas can then be done by coding the origin of the contribution into the delay to the associative area. The time order code could be embedded in oscillatory brain rhythms in order to have a common time reference (e.g.. gamma rhythms, which have been proposed to underlie the binding of different features into a single perceptual entity [33] [26]). Such oscillatory behaviour could be obtained with re-entrant connections [29] and the rank order code could synchronized with it and be coded on the crest of each oscillation.

VIII. Conclusions

In this paper we have presented a study on how rank order coding and polychronization can be used to store and decode information. The system proposed is able to store and recall codes in a polychronous sparse distributed memory layer. We have successfully tested the discrimination power of this model with rank order codes representing significance or intensity of a stimulus in the spike timings, even in the presence of noise. The empirical evidence collected encourage us to research a more formal definition of this model, enrich it and scale it up. Finally the model could be tested in interactive tasks, embodying the system in a virtual or real environment to explore its further capabilities.

Acknowledgement

We would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for support of this research through grant EP/4015740/1.

References

[1] http://neuron.duke.edu/.

[2] Guoqiang Bi and Muming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, 18(24):10464–10472, 1998.

[3] Frances S. Chance and L. F. Abbott. Divisive inhibition in recurrent networks. *Network*, 11:119–129, 2000.

[4] Bruederle D., Muller E., Davison A., Muller E., Schemmel J., and Meier K. Establishing a novel modeling tool: A python-based interface for a neuromorphic hardware system. *Frontiers in Neuroinformatics*, 3:17:doi:10.3389/neuro.11.017.2009, 2009.

[5] A.P. Davison, D. Bruderle, J.M. Eppler, J. Kremkow, E. Muller, D.A. Pecevski, L. Perrinet, and P. Yger. Pynn: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2, 2008.

[6] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 1st edition, December 2001.

[7] A Delorme. Face identification using one spike per neuron: resistance to image degradations. *Neural Networks*, 14(6-7):795–803, 2001.

[8] A. Delorme. Early cortical orientation selectivity: How fast inhibition decodes the order of spike latencies. *Journal of Computational Neuroscience*, 15:357–365, 2003. 10.1023/A:1027420012134.

[9] Stephen B. Furber, G. Brown, Joy Bose, J. Mike Cumpstey, P. Marshall, and Jonathan L. Shapiro. Sparse distributed memory using rank-order neural codes. *IEEE Transactions on Neural Networks*, 18(3):648–659, 2007.

[10] Francesco Galluppi, Alexander D. Rast, Sergio Davies, and Steve Furber. A general-purpose model translation system for a universal neural chip. In *ICONIP (1)*, pages 58–65, 2010.

[11] M. L. Hines and N. T. Carnevale. The NEURON Simulation Environment. *Neural Computation*, 9(6):1179–1209, August 1997.

[12] A. L. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Physiol. London*, 117:500–544, 1952. article.

[13] Eugene M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(2):245–282, February 2006.

[14] Eugene M. Izhikevich. Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling. *Cerebral Cortex*, 17(10):2443–2452, October 2007.

[15] Eugene M. Izhikevich, Joe A. Gally, and Gerald M. Edelman. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, 14:933–944, 2004.

[16] Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, Cambridge, MA, USA, 1988.

[17] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, December 1997.

[18] W. Maass. Neural computation with winner-take-all as the only nonlinear operation. 12:293–299, 2000.

[19] W. Maass, G. Schnitger, and E. Sontag. On the computational power of sigmoid versus boolean threshold circuits. In *32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 767–776, 1991. inproceedings.

[20] Wolfgang Maass, Thomas Natschl?ger, Technische Universitaet Graz, and Henry Markram. On the computational power of circuits of spiking neurons. *J. of Physiology (Paris*, 2005, 2003.

[21] B. Q. Mao, F. Hamzei-Sichani, D. Aronov, R. C. Froemke, and R. Yuste. Dynamics of spontaneous activity in neocortical slices. *Neuron*, 32(5):883–898, Dec 2001.

[22] E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001.

[23] Filip Ponulak. Supervised learning in spiking neural networks. *The Neuromorphic Engineer*.

[24] Basabdatta Sen and Steve Furber. Information recovery from rank-order encoded images. In *Proceedings: Workshop on Biologically Inspired Information Fusion*, 2006.

[25] Basabdatta Sen and Steve Furber. Evaluating rank-order code performance using a biologically-derived retinal model. In *Proceedings of the 2009 international joint conference on Neural Networks*, IJCNN'09, pages 1835–1842, Piscataway, NJ, USA, 2009. IEEE Press.

[26] W Singer and CM Gray. Visual Feature Integration and the Temporal Correlation Hypothesis. *Annual Review of Neuroscience*, 18:555–586, 1995.

[27] Sen Song, Kenneth D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919 – 926, 2000.

[28] O. Sporns, N. Almássy, and G. M. Edelman. Plasticity in value systems and its role in adaptive behavior. *Adaptive Behavior*, 8(2):129–148, February 2000.

[29] O. Sporns, J. A. Gally, G. N. Reeke, and G. M. Edelman. Reentrant signaling among simulated neuronal groups leads to coherency in their oscillatory activity. *Proc Natl Acad Sci U S A*, 86(18):7265–7269, Sep 1989.

[30] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[31] Simon Thorpe and Jacques Gautrais. Rank order coding. In *CNS '97: Proceedings of the sixth annual conference on Computational neuroscience : trends in research, 1998*, pages 113–118, New York, NY, USA, 1998. Plenum Press.

[32] Simon J. Thorpe, Arnaud Delorme, and Rufin van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6-7):715–725, 2001.

[33] G. Tononi, O. Sporns, and G. M. Edelman. Reentry and the problem of integrating multiple cortical areas: simulation of dynamic integration in the visual system. *Cereb Cortex*, 2(4):310–335, 1992.

[34] G. Tononi, O. Sporns, and GM Edelman. *Proceedings of the National Academy of Sciences*, 91(11):5033–5037, 1994.

[35] Rufin Van Rullen, Jacques Gautrais, Arnaud Delorme, and Simon Thorpe. Face processing using one spike per neurone. *Biosystems*, 48(1-3):229–239, November 1998.