Delay Learning and Polychronization for Reservoir Computing

Hélène Paugam-Moisy [a,*] Régis Martinez [a] and Samy Bengio [b]

[a] *LIRIS, UMR CNRS 5205, Université Lyon 2, France*

[b] *Google, 1600 Amphitheatre Pkwy, B1350-138B, Mountain View, CA 94043, USA*

Abstract

We propose a multi-timescale learning rule for spiking neuron networks, in the line of the recently emerging field of reservoir computing. The reservoir is a network model of spiking neurons, with random topology and driven by STDP (Spike-Time-Dependent Plasticity), a temporal Hebbian unsupervised learning mode, biologically observed. The model is further driven by a supervised learning algorithm, based on a margin criterion, that affects the synaptic delays linking the network to the readout neurons, with classification as a goal task. The network processing and the resulting performance can be explained by the concept of polychronization, proposed by Izhikevich (2006, Neural Computation, 18:2), on physiological grounds. The model emphasizes that polychronization can be used as a tool for exploiting the computational power of synaptic delays and for monitoring the topology and activity of a spiking neuron network.

Key words: Reservoir Computing, Spiking Neuron Network, Synaptic Plasticity, STDP, Polychronization, Programmable Delay, Margin Criterion, Classification

---

* LIRIS, UMR CNRS 5205, Bât. C, Université Lumière Lyon 2, 5 avenue Pierre Mendès France, F-69676 Bron cedex, France

Email addresses: `hpaugam@liris.cnrs.fr` (Hélène Paugam-Moisy), `regis.martinez@liris.cnrs.fr` (Régis Martinez), `bengio@google.com` (Samy Bengio).

URLs: `http://liris.cnrs.fr/membres?idn=hpaugam` (Hélène Paugam-Moisy), `http://liris.cnrs.fr/membres?idn=rmartine` (Régis Martinez), `http://bengio.abracadoudou.com` (Samy Bengio).

1 Introduction

1.1 Reservoir Computing

Reservoir Computing (RC) recently appeared [38,17] as a generic name for designing a new research stream including mainly Echo State Networks (ESNs) [15,16], Liquid State Machines (LSMs) [25], and a few other models like Backpropagation Decorrelation (BPDC) [42]. Although they have been discovered independently, the algorithms share common features and carry many highly challenging ideas toward a new computational paradigm of neural networks. The central specification is a large, distributed, nonlinear recurrent network, the so-called "reservoir", with trainable output connections, devoted to reading out the internal states induced in the reservoir by input patterns. Usually, the internal connections are sparse and their weights are kept fixed. According to the model, the reservoir can be composed of different types of neurons, e.g. linear units, sigmoid neurons, threshold gates or spiking neurons (e.g. LIF [1]), as far as the internal network behaves like a non linear dynamical system. In most models, simple learning rules, such as linear regression or recursive least mean squares are applied to readout neurons only. In the editorial of the special issue of Neural Networks [17], several directions for further research have been pointed out, among them is "the development of practical methods to optimize a reservoir toward the task at hand". In the same article, RC is presented as belonging to the "family of versatile basic computational metaphors with a clear biological footing". The model we developed [34] is clearly based on similar principles: A network of spiking neurons, sparsely connected, without pre-imposed topology, and output neurons, with adaptable connections.

As stated in [38], the concept that is considered to be a main advantage of RC is to use a fixed randomly connected network as reservoir, without training burden. Recent work also propose to add an unsupervised reservoir adaptation through various forms of synaptic plasticity such as Spike-Time-Dependent Plasticity [31] or Intrinsic Plasticity [47,50] (see [21] for a comparative study). However, many articles point out the difficulty to get a suitable reservoir w.r.t. a given task. According to the cognitive metaphor, the brain may be viewed as a huge reservoir able to process a wide variety of different tasks, but synaptic connections are not kept fixed in the brain: Very fast adaptation processes compete with long term memory mechanisms [18]. Although the mystery of memory and cognitive processing is far from being solved, recent advances in neuroscience [2,28,41] help to get new inspiration for conceiving computational models. Our learning rule for readout neurons is justified by

[1] LIF = Leaky Integrate and Fire

the appealing notion of polychronization [14] from which Izhikevich derives an explanation for a theoretically "infinite" capacity of memorizing in the brain. We based our method to adapt the reservoir to the current task on an implementation of synaptic plasticity inside a Spiking Neuron Network (SNN) and on the exploitation of the polychronization concept.

1.2 Spiking Neuron Networks

A common thought that interactions between neurons are governed by their mean firing rates has been the basis of most traditional artificial neural network models. Since the end of the 90's, there is a growing evidence, both in neuroscience and computer science, that precise timing of spike firing is a central feature in cognitive processing. SNNs derive their strength and interest from an accurate modeling of synaptic interactions between neurons, taking into account the time of spike firing. Many biological arguments, as well as theoretical results (e.g. [22,37,40]) converge to establish that SNNs are potentially more powerful than traditional artificial neural networks. However, discovering efficient learning rules adapted to SNNs is still a hot topic. For the last ten years, solutions were proposed for emulating classic learning rules in SNNs [24,30,4], by means of drastic simplifications that often resulted in losing precious features of firing time based computing. As an alternative, various researchers have proposed different ways to exploit recent advances in neuroscience about synaptic plasticity [1], especially IP [2] [10,9] or STDP [3] [28,19], that is usually presented as the Hebb rule, revisited in the context of temporal coding. A current trend is to propose computational justifications for plasticity-based learning rules, in terms of entropy minimization [5] as well as log-likelihood [35] or mutual information maximization [8,46,7]. However, since STDP is a local unsupervised rule for adapting the weights of connections, such a synaptic plasticity is not efficient enough for controlling the behavior of an SNN in the context of a given task. Hence we propose to couple STDP with another learning rule, acting at a different time scale.

1.3 Multi-timescale Learning

We propose to name "multi-timescale learning" a learning rule combining at least two adaptation algorithms, at different time scales. For instance, synaptic plasticity, modifying the weights locally, in the millisecond time range, can be coupled with a slower overall adaptation rule, such as reinforcement learning

[2] IP = Intrinsic Plasticity

[3] STDP = Spike-Time-Dependent Plasticity

driven by an evolutionary algorithm, like in [29], or a supervised learning algorithm, for classification purpose, as developed in the present article.

The multi-timescale learning rule we propose is motivated by two main ideas:

- **Delay adaptation:** Several complexity analyzes of SNNs have proved the interest of programmable delays for computational power [22,37] and learnability [26,27,23]. Although axonal transmission delays do not vary continually in the brain, a wide range of delay values have been observed.
- **Polychronization:** In [14] Izhikevich pointed out the activation of polychronous groups, based on the variability of transmission delays inside an STDP-driven set of neurons (see Section 5 for details), and proposed that the emergence of several polychronous groups, with persistent activation, could represent a stimulation pattern.

Our multi-timescale learning rule for reservoir computing is comprised of STDP, modifying the weights inside the reservoir, and a supervised adaptation of axonal transmission delays toward readout neurons coding, via their times of spike firing, for different classes. Without loss of generality, the model is mainly presented in the two-class version. The basic idea is to adapt the output delays in order to enhance the influence of the polychronous groups activated by a given pattern toward the target output neuron, and to decrease the influence toward the non-target neuron. A margin criterion is applied, via a stochastic iterative learning process, for strengthening the separation between the spike-timing of the readout (output) neurons. This idea fits in the similarity that has been recently proposed [38,17] between RC and SVM [4], where the reservoir is compared to the high-dimensional feature space resulting from a kernel transformation. In our algorithm, like in the machine learning literature, the application of a margin criterion is justified by the fact that maximizing a margin between the positive and the negative class yields better expected generalization performance [48].

We point out that polychronization can be considered as a tool for adapting synaptic delays properly, thus exploiting their computational power, and for observing the network topology and activity.

The outline of the papers goes as follows: Section 2 describes the model of spiking neuron network (SNN) for reservoir computing (RC); Section 3 defines the multi-scale learning mechanism; experiments on classification tasks are presented in Section 4; Section 5 explains the notion of polychronization and Section 6 studies the internal dynamics of the reservoir.

[4] SVM = Support Vector Machine

2 Network Architecture

The reservoir is a set of M neurons (internal network), interfaced with a layer of K input cells and C readout cells, one for each class (Figure 1). The network is fed by input vectors of real numbers, represented by spikes in temporal coding: The higher the value, the earlier the spike fires toward the reservoir. For clarity in experiments, successive inputs are presented in large temporal intervals, without overlapping input spike firing from a pattern to the next. The index of the output cell firing first in this temporal interval provides the class number as an answer of the network to the input pattern.

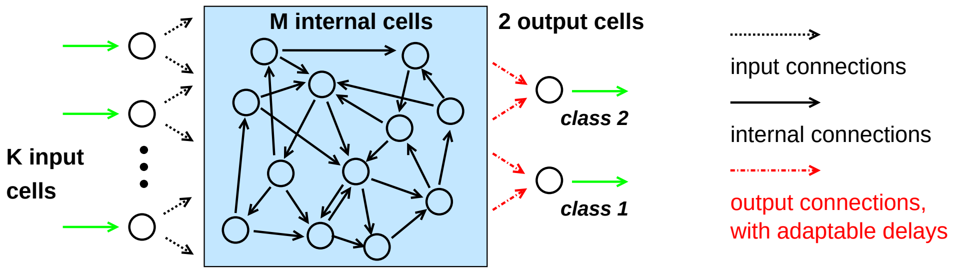


Fig. 1. Architecture of the spiking neuron network. The reservoir is the internal network (the colored square) and green (light grey) links represent the interface with environment. The network is presented for $C = 2$ classes.

Each cell is a spiking neuron (Section 2.1). Each synaptic connection, from neuron N_i to neuron N_j, is defined by two parameters: A weight w_{ij} and an axonal transmission delay d_{ij}. The reservoir is composed of 80% excitatory neurons and 20% inhibitory neurons, in accordance with the ratio observed in the mammalian cortex [6]. The internal connectivity is random and sparse, with probability P_{rsv} for a connection to link N_i to N_j, for all $(i, j) \in \{1, ..., M\}^2$. For pattern stimulation, the input cells are connected to the internal cells with probability P_{in}. The connection parameters are tuned so that the input cells forward spikes toward internal cells according to the temporal pattern defined by the input stimulus (see Section 2.2). For class detection, the output neurons are fully connected to each internal cell ($P_{out} = 1$).

2.1 Neuron Model

The neuron model is an SRM_0 (zero'th order "Spike Response Model"), as defined by Gerstner [12], where the state of a neuron N_j is dependent on its last spike time $t_j^{(f)}$ only. The next firing time of N_j is governed by its membrane potential $u_j(t)$, in millivolts, and its threshold $\theta_j(t)$. Both variables depend on the last firing times of the neurons N_i belonging to the set Γ_j of neurons pre-synaptic to N_j:

$$u_j(t) = \underbrace{\eta(t - t_j^{(f)})}_{threshold\ kernel} + \sum_{i \in \Gamma_j} w_{ij} \underbrace{\epsilon(t - t_i^{(f)} - d_{ij})}_{potential\ kernel} + u_{rest} \tag{1}$$

$$\text{firing condition for } N_j \qquad u_j(t) \geq \vartheta \text{ with } u_j'(t) > 0 \implies t_j^{(f+1)} = t \tag{2}$$

The potential kernel is modelled by a Dirac increase in 0, followed by an exponential decrease, from value u_{max} in 0^+ toward 0, with a time constant τ_m, in milliseconds:

$$\epsilon(s) = u_{max} \mathcal{H}(s) e^{\left(-\frac{s}{\tau_m}\right)} \tag{3}$$

where $\mathcal{H}$ is the Heaviside function. In Equation (2) the value w_{ij} is a positive factor for excitatory weights and a negative one for inhibitory weights.

The firing threshold ϑ is set to a fixed negative value (e.g. $\vartheta = -50\ mV$) and the threshold kernel simulates an absolute refractory period τ_{abs}, when the neuron cannot fire again, followed by a reset to the resting potential u_{rest}, lower than ϑ (e.g. $u_{rest} = -65\ mV$). The relative refractory period is not simulated in our model. The simulation is computed in discrete time with 1 ms time steps. Time steps 0.1 ms long have been tested also (Section 4.2). The variables of neuron N_j are updated at each new incoming spike (event-driven programming), which is sufficient for computational purpose.

2.2 Weights and Delays

Synaptic plasticity (see Section 3.1) is applied to the weights of internal cells only. Starting from initial values w such that $\|w\| = 0.5$, internal weights vary, under STDP, in the range $[0, 1]$ (excitatory) or $[-1, 0]$ (inhibitory). The weights of connections from input layer to internal network are kept fixed, all of them excitatory, with a value w_{in} strong enough to induce immediate spike firing inside the reservoir (e.g. $w_{in} = 3$ in experiments, see Section 4). The connections from internal network to output neurons are excitatory and the output weights are fixed to the intermediate value $w_{out} = 0.5$. In principle, STDP can be applied also to the output weights (optional in our program) but no improvement has been observed. Hence, for saving computational cost, the readout learning rule has been further restricted to an adaptation of the synaptic delays (Section 3.2).

Neuroscience experiments [44,45] give evidence to the variability of transmission delay values, from $0.1ms$ to $44ms$. In the present model, the delays d_{ij} take integer values, randomly chosen in $\{d_{min}, \ldots, d_{max}\}$, rounded to the nearest millisecond, both in the internal network and toward readout neurons. The delays from input layer to internal network have a zero value, for an immediate transmission of input information.

A synaptic plasticity rule could be applied to delay learning, as well as to weight learning, but the biological plausibility of such a plasticity is not yet so clear in neuroscience [39]. Moreover our purpose is to exploit this stage of the learning rule for making easier the adaptation of the reservoir to a given task. Hence we do not apply synaptic plasticity to delays, but we switch to machine learning in designing a supervised mechanism, based on a margin criterion, for adapting the output delays to the reservoir computation in order to extract the relevant information.

3 Learning Mechanisms

In the model, the multi-timescale learning rule is based on two concurrent mechanisms: A local unsupervised learning of weights by STDP, operating in the millisecond range, at each new incoming spike time t_{pre} or outgoing spike time t_{post}, and a supervised learning of output delays, operating in the range of 100 ms, at each pattern presentation.

3.1 Synaptic Plasticity

The weight w_{ij} of a synapse from neuron N_i to neuron N_j is adapted by STDP, a form of synaptic plasticity based on the respective order of pre- and post-synaptic firing times. For excitatory synapses, if a causal order (pre- just before post-) is respected, then the strength of the connection is increased. Conversely the weight is decreased if the pre-synaptic spike arrives at neuron N_j just after a post-synaptic firing, and has probably no effect, due to the refractory period of N_j. For inhibitory synapses, only a temporal proximity leads to a weight increase, without causal effect. Temporal windows, inspired from neurophysiological experiments by Bi & Poo [3], help to calculate the weight modification ΔW as a function of the time difference $\Delta t = t_{post} - t_{pre} = t_j^{(f)} - (t_i^{(f)} + d_{ij})$ as can be computed at the level of neuron N_j.

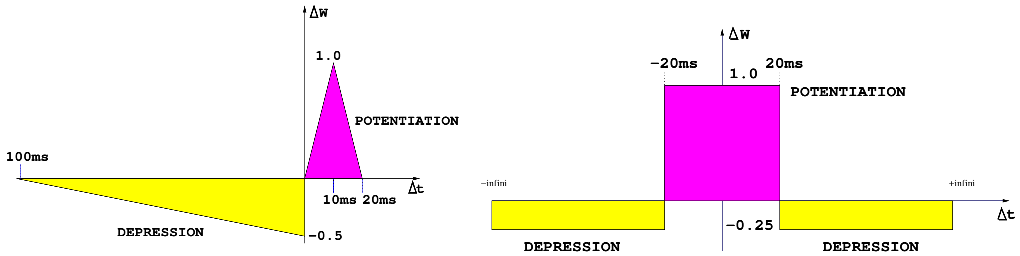


Fig. 2. Asymmetrical STDP temporal window for excitatory (left) and symmetrical STDP temporal window for inhibitory (right) synapse adaptation (from [29]).

For updating excitatory synapses as well as inhibitory synapses, a similar principle is applied, and only the temporal windows differ (see Figure 2). For updating excitatory synapses, we adopt the model of Nowotny [32] with an asymmetrical shape of temporal window. For inhibitory synapses, weight updating is based on a correlation of spikes, without influence of the temporal order, as proposed in [51].

Following [36], in order to avoid a saturation of the weights to the extremal values $w_{min} = 0$ and $w_{max} = 1$ (excitatory) or $w_{max} = -1$ (inhibitory), we apply a multiplicative learning rule, as stated in Equation (4), where α is a positive learning rate. In our experiments: $\alpha = \alpha_{exc} = \alpha_{inh} = 0.1$. For excitatory synapses the sign of ΔW is the sign of Δt. For inhibitory synapses $\Delta W > 0$ iff $\mid \Delta t \mid < 20$.

$$\begin{aligned} &\text{if } \Delta W \leq 0 \text{ depreciate the weight: } w_{ij} \leftarrow w_{ij} + \alpha * (w_{ij} - w_{min}) * \Delta W \\ &\text{if } \Delta W \geq 0 \text{ potentiate the weight: } w_{ij} \leftarrow w_{ij} + \alpha * (w_{max} - w_{ij}) * \Delta W \end{aligned} \tag{4}$$

STDP is usually applied with an additive rule for weight modification and many authors observe a resulting bimodal distribution of weights, with an effect of saturation toward the extremal values. In [21] Lazar et al. propose to couple IP (Intrinsic Plasticity) with STDP and show that the effect of saturation is reduced. We obtain a similar result with a multiplicative rule (see Section 4.1.3), but at a lower computational cost.

3.2 Delay Adaptation Algorithm

The refractory period of the readout neurons has been set to a value τ_{abs}^{out}, large enough to trigger at most one output firing for each neuron, inside a temporal window of T ms dedicated to an input pattern presentation.

The goal of the supervised learning mechanism is to modify the delays from active internal neurons to readout neurons in such a way that the output neuron corresponding to the target class fires before the one corresponding to the non-target class. Moreover, we intend to maximize the margin between the positive and the negative class. More formally, the aim is to minimize the following criterion:

$$C = \sum_{p \in \text{class1}} |t_1(p) - t_2(p) + \mu|_+ + \sum_{p \in \text{class2}} |t_2(p) - t_1(p) + \mu|_+ \tag{5}$$

where $t_i(p)$ represents the firing time of readout neuron Out_i after the presentation of input pattern p, μ represents the minimum delay margin we want to enforce between the two firing times, and $|z|_+ = \max(0, z)$. The margin constant μ is a hyper-parameter of the model and can be tuned according to the task at hand. Convenient values are a few milliseconds, e.g. $\mu = 5$ or $\mu = 8$ in experiments (Section 4).

In order to minimize the criterion (5), we adopt a stochastic training approach, iterating a delay adaptation loop, as described in Algorithm 1. We define a *triggering* *connection* as a connection that carries an incoming spike responsible for a post-synaptic spike firing at the impact time. Due to the integer values of delays and the discrete time steps for computation, it may occur that several triggering connections accumulate their activities at a given iteration. In such a case, we choose only one among these candidate connections for delay adaptation.

Algorithm 1 (two-class):

```
repeat
  for each example X = (p, class) of the database
  {   present the input pattern p;
      define the target output neuron according to class;
      if ( the target output neuron fires less than μ ms before
          the non-target output neuron, or fires after it )
      then
      {   select one triggering connection of the target output
          neuron and decrement its delay (−1 ms), except if
          d_min is reached already;
          select one triggering connection of the non-target
          output neuron and increment its delay (+1 ms), except
          if d_max is reached already;
      }
  }
until a given maximum learning time is over.
```

As an illustration, let us consider an input pattern belonging to class 2. Hence we want output neuron Out_2 to fire at least μ milliseconds before output neuron Out_1. Figure 3 shows the variation through time of the membrane potential of the two readout neurons. Note that, without loss of generality, the curves of exponential decrease have been simplified into straight oblique lines, to represent variations of $u(t)$. The effect of one iteration of delay learning on the respective firing times is depicted from the left graphic (step k) to the right one (step $k+1$). At step k, the difference between firing times of the two output neurons is lower than μ, whereas at step $k+1$ the pattern is well classified, with respect to the margin.

Although the context is not always as auspicious as in Figure 3, even so, at each iteration where a delay adaptation occurs, the probability of an error in the next answer to a similar input has decreased. Actually, under the hypothesis that the recent history of the membrane potential is similar between the current and the next presentation of a pattern p (verified condition in Figure 3), the increment of the delay of the non-target neuron leads to a later firing of Out_1 with probability 1. Under similar conditions, if the triggering connection of the target neuron is alone, the decrement of its delay causes a 1 ms earlier spike, thus incoming in addition to a higher value of the membrane potential as far as the time constant τ_m generates an exponential decrease less than 0.5 (equal to the contribution of w_{out}) in a 1 ms time range, hence the readout neuron Out_1 fires earlier, with probability 1. The only hazardous situation occurs in case of multiple triggering connections,

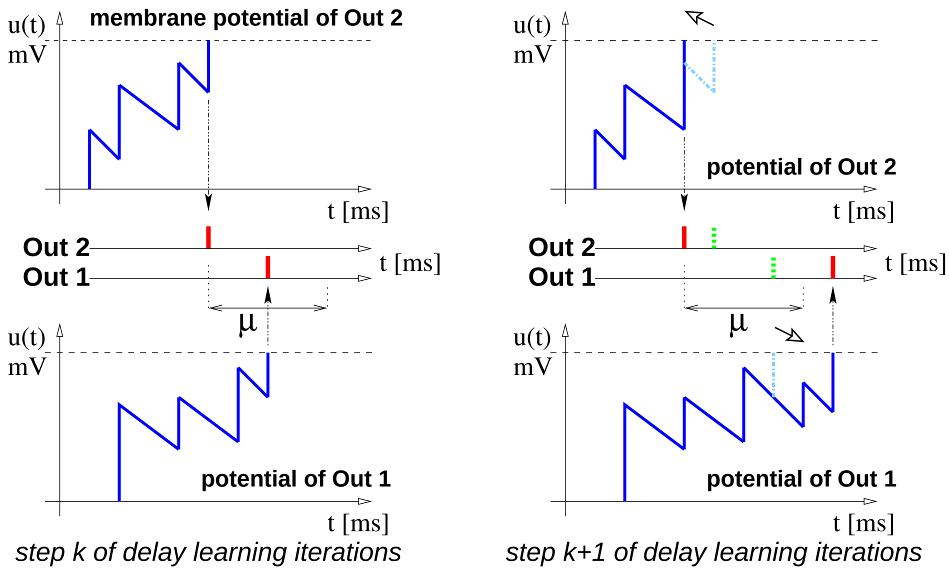


Fig. 3. An example illustrating the effect of one iteration of the delay learning algorithm on the firing times of the two readout neurons.

where configurations exist for later firing of Out_1, but with a probability close to 0, since the sparse connectivity of the reservoir induces a low internal overall spiking activity (usually close to 0.1 in experimental measurements). Finally, the hypothesis of membrane potential history conservation is not highly constraining, since the action of STDP has the effect of reinforcing the weights of the connections, in the reservoir, that are responsible for relevant information transmission. Therefore, as the learning process goes through time, spike-timing patterns become very stable, as could be observed experimentally (see raster plots in Section 4.1.1).

3.3 Extension to Multi-Class Discrimination

The multi-timescale learning rule can be extended to multi-class discrimination. The network architecture is similar (Figure 1), except that the output layer is composed of C readout neurons, one for each of the C classes. The response of the network to an input pattern p is the index of the first firing output neuron. Whereas synaptic plasticity is unchanged, the delay adaptation algorithm has to be modified and several options arise, especially concerning the action on one or several non-target neurons. We propose to apply Algorithm 2.

A variant could be to increment the delays of all the first firing non-target output neurons. Since the performance improvement is not clear, in experiments the advantage has been given to Algorithm 2, i.e. with at most one delay adaptation at each iteration, in order to save computational cost.

Algorithm 2 (multi-class):

```
repeat
  for each example X = (p, class) of the database
  {   present the input pattern p;
      define the target output neuron according to class;
      if ( the target output neuron fires less than μ ms before
           the second firing time among non-target output neurons,
           or fires later than one or several non-target neurons )
      then
      {    randomly select one triggering connection of the target
           output neuron and decrement its delay (−1 ms), except
           if d_min is reached already;
           randomly select one neuron among all the non-target
           readouts that fired in first;
           for this neuron, randomly select one triggering
           connection and increment its delay (+1 ms), except if
           d_min is reached already;
      }
  }
until a given maximum learning time is over.
```

4 Experiments

A first set of experiments has been performed on a pair of very simple patterns, borrowed from Figure 12 of [14], in order to understand how the network processes and to verify the prior hypotheses we formulated on the model behavior, in particular the emergence of polychronous groups with persistent activity in response to a specific input pattern. A second set of experiments is then presented on the USPS handwritten digit database, in order to validate the model ability to classify real-world, complex, large scale data. Since the main purpose is to study the internal behavior of the network and the concept of polychronization, experiments have been performed mainly in the two-class case, even with the USPS database.

4.1 Two-Class Discrimination on Izhikevich's Patterns

The task consists in discriminating two diagonal bars, in opposite directions. The patterns are presented inside successive time windows of length $T = 20\ ms$. For this task, the neuron constants and network hyper-parameters have been set as follows:

Network architecture:
$K = 10$ input neurons
$M = 100$ neurons in the reservoir.

Spiking neuron model:
$u_{max} = 8\ mV$ for the membrane potential exponential decrease
$\tau_m = 3\ ms$ for time constant of the membrane potential exponential decrease
$\vartheta = -50\ mV$ for the neuron threshold
$\tau_{abs} = 7\ ms$ for the absolute refractory period
$u_{rest} = -65\ mV$ for the resting potential

Connectivity parameters:
$P_{in} = 0.1$ connectivity probability from input neurons toward the reservoir
$P_{rsv} = 0.3$ connectivity probability inside the reservoir
$w_{in} = 3$, fixed value of weights from input neurons to the reservoir
$w_{out} = 0.5$, fixed value of weights from reservoir to output neurons
$d_{min} = 1$, minimum delay value, in the reservoir and toward the readouts
$d_{max} = 20$, maximum delay value, in the reservoir and toward the readouts

Delay adaptation parameters:
$\tau_{abs}^{out} = 80\ ms$ for the refractory period of readout neurons
$\mu = 5$ for the margin constant

At any time (initialization, learning and generalization phases), the complete cartography of the network activity can be observed on a spike raster plot presenting all the firing times of all neurons (see for instance Figure 4): Neuron index with respect to time (in ms) with $K = 10$ input neurons (bottom), followed by $M = 100$ internal neurons, including 20 inhibitory neurons, in blue (light grey). Firing times of the two output neurons are isolated at the top.

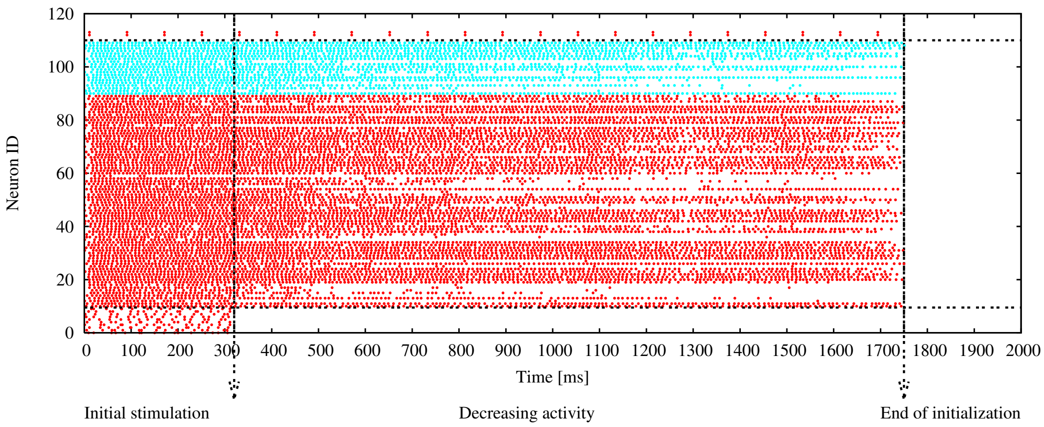


Fig. 4. Spike raster plot, for initial random stimulation, from time 0 to 2000 ms.

A run starts by an initialization phase. All connections in the reservoir are initialized with weights $w_{ij} = 0.5$ (excitatory) or $w_{ij} = -0.5$ (inhibitory). Gaussian noise is presented in input during the first 300 ms, thus generating a high disordered activity in the reservoir (Figure 4) and frequent weight updating. Output neurons spike simultaneously, as soon as their refractory period ends. Due to STDP, the internal activity slowly decreases until complete silence around 1750 ms.

4.1.1 Learning

Afterwards, a learning phase is run, between times T_{L1} and T_{L2}. Figure 5 presents two time slices of a learning run, with successive alternated presentations of the two input patterns that represent examples for class 1 and class 2 respectively. As can be observed, the internal network activity quickly decreases and then stabilizes on a persistent alternative between two different spike-timing patterns (lasting slightly longer than the time range of pattern presentation), one for each class. The learning performance is a 100% success rate, even in experiments where the patterns to be learned are presented in random order.

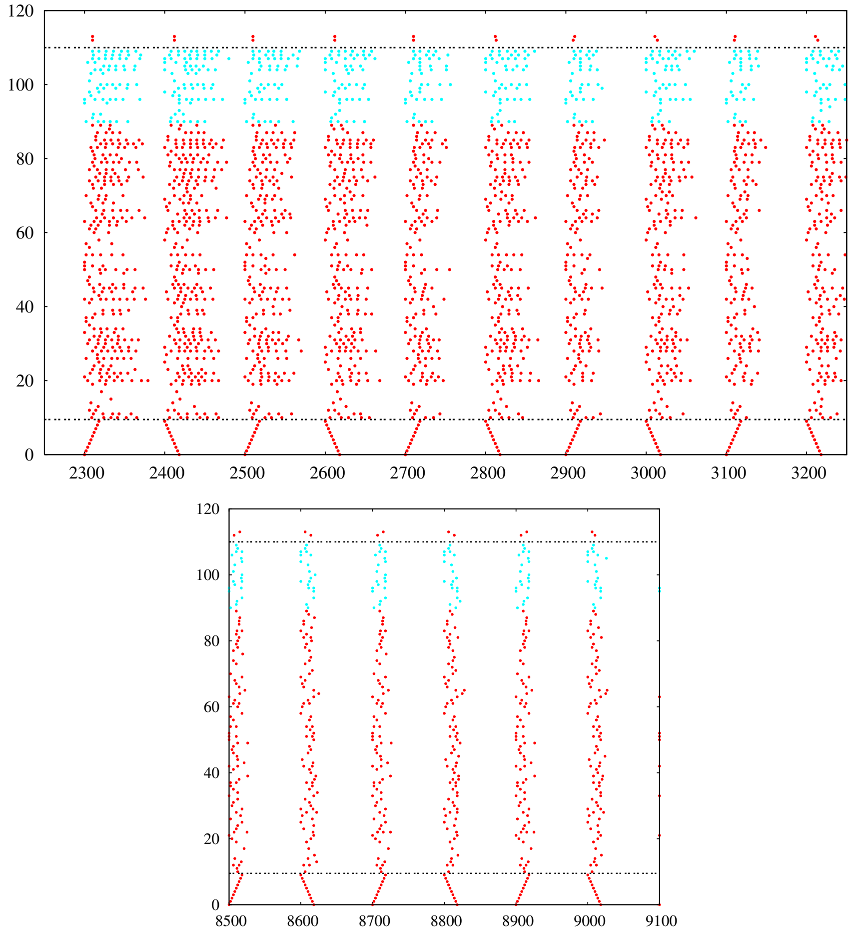


Fig. 5. Spike raster plots, for two time slices of a learning run, one just after T_{L1} (top) and the other a long time after activity has stabilized, until T_{L2} (bottom).

The evolution of the firing times of the two output neurons reflects the application of the delay adaptation algorithm. Starting from simultaneous firing, they slightly dissociate their responses, from a pattern presentation to the next, according to the class corresponding to the input (top frame, Figure 5). In the bottom frame of the figure, the time interval separating the two output spikes has become larger, due to delay adaptation, and is stable, since the margin μ has been reached. The internal activity is quite invariant, except for occasional differences due to the still running STDP adaptation of weights. This point will be discussed later (Sections 4.1.3 and

6).

4.1.2 Generalization

Finally between T_{G1} and T_{G2} a generalization phase is run with noisy patterns: Each spike time occurs at $t \pm \eta$ where t is the firing time of the corresponding input neuron for the example pattern of the same class and η is some uniform noise. In Figure 6, two noise levels can be compared. Although the internal network activity is clearly disrupted, the classification performance remains good: Average success rate, on 100 noisy patterns of each class, is 96% for $\eta = 4$, when noisy patterns are presented alternatively, class 2 after class 1, and still 81% for $\eta = 8$, where the input patterns are hard to discriminate by a human observer. We observed a slight effect of sequence learning: Only 91% and 73% success respectively for $\eta = 4$ and $\eta = 8$, when class 1 and class 2 are presented in random order.

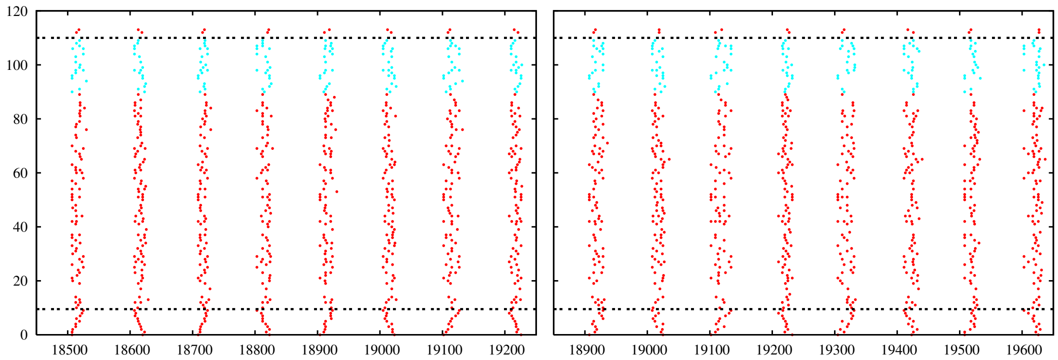


Fig. 6. Spike raster plots, for two series of noisy patterns: $\eta = 4$ (top) and 8 (bottom).

We observe that the obtained margin between the two output firing times can be higher or lower than μ. For each pattern, this margin could be exploited as a confidence measure over the network answer. Moreover, most of the non-successful cases are due to simultaneous firing of the two output neurons (in Figure 6, only one wrong order near the left of 18800 ms). Such ambiguous responses can be considered as "non-answers", and could lead to define a subset of rejected patterns. Wrong order output spike-firing patterns are seldom, which attest the robustness of the learning algorithm.

The performance is increased and the sequence effect disappears when the margin constant is set to a higher value. For $\mu = 8$ instead of $\mu = 5$, the generalization success rate reaches 100% for $\eta = 4$ and 90% for $\eta = 8$, for both an alternate or a random presentation of the patterns. In the latter case, the error rate is only 0.3% and the 9.7% remaining cases are "non-answers". This phenomenon could be used as a criterion for tuning the margin hyper-parameter.

4.1.3 Weight Distributions

In order to illustrate the weight adaptation that occurs in the reservoir, Figures 7 and 8 show respectively the distribution of excitatory and inhibitory weights (in absolute value) quantized into 10 uniform segments of 0.1 and captured at different times. The distribution at time 0 is not shown, as all the $\mid w_{ij} \mid$ were initialized to 0.5. Afterwards, it can be checked that weights are widely distributed in the range $[w_{min}, w_{max}]$. First, at the end of initialization phase, excitatory weights (Figure 7) tend to be Gaussian around the original distribution (time 300 and 2000). We have measured that the average amount of time between two spikes during the first 1700 ms corresponds to 8 ms. In the excitatory STDP temporal window (Figure 2, left), $|\Delta W|$ in the range of 8 ms is comparable at both sides of 0, and thus explains this Gaussian redistribution. Then, during the learning phase, weights uniformly distribute, mainly from 0 to 0.7, for instance at time 4000. Around 10000 ms, an equilibrium is reached since strong variations of weights no longer occur under the influence of STDP. It can be thought that the causal order of firing has been captured inside the network. The distribution of weight values is approximately 50% very close to 0, other weights being decreasingly distributed from 0.1 to 0.7.

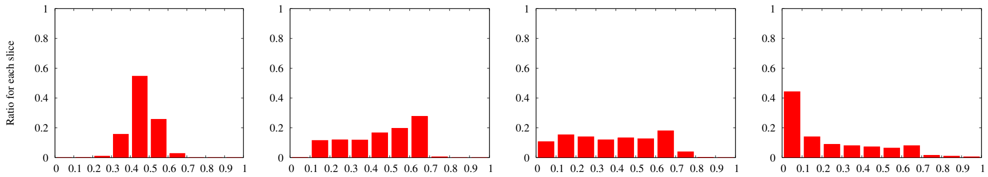


Fig. 7. Excitatory weights distribution at time 300, 2000, 4000, 10000 ms.

Let us now consider inhibitory weights in Figure 8. As the initial internal activity is strong, the weights are modified in a very short time range. Indeed, looking at time 300 (Figure 8, left) we see that weights have already nearly all migrated to an extremal value (close to -1). This surprising violent migration can as well be explained by the inhibitory STDP function, where close spikes in an inhibitory synapse produce a strong weight potentiation (see Figure 2, right). A high activity strongly potentiates the inhibitory synapses that, in turn, slow down the activity, thus playing a regulatory role. After the initial stimulation stopped, weights begin to redistribute as the reservoir activity slows down. From then on, weight distribution has reached a state that slightly evolves, until time 10000, and stays very stable until time 17000 (end of learning phase).

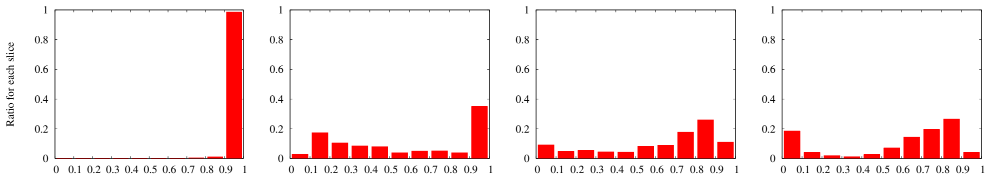


Fig. 8. Distribution of $\mid w_{ij} \mid$ for inhibitory weights at time 300, 2000, 4000, 10000 ms.

Note that, due to the multiplicative [36] application of STDP temporal windows (Section 3.1), the weights are never strictly equal to w_{min} or w_{max}. Experiments show that they do not saturate toward extremal values. This observation confirms the interest of multiplicative STDP: The effect on the weight distribution is comparable to the result of combining IP (Intrinsic Plasticity) and classic STDP, that has been proved to enhance the performance and the network stability [21].

In [15] Jaeger claims that the spectral radius of the weight matrix must be smaller than 1. This point has been widely discussed and confirmed by studies on the network dynamics proving that a spectral radius close to 1 is an optimal value. However we share the opinion of Verstraeten et al. [49] who claim that "for spiking neurons it has no influence at all". In [43] Steil shows that a learning rule based on IP has the effect to expand the eigenvalues away from the center of the unit disk. On few measurements, we observed a converse effect with our learning rule and with spectral radii higher than 1, e.g. $\lambda = 8.7$ for the initial weight matrix and $\lambda = 2.3$ after a learning phase of 20000 ms. This point remains to be more deeply investigated, both through more experimental measurements and a theoretical study.

4.2 OCR on the USPS Database

The patterns of the USPS dataset [5] [13] consist of 256 dimensional vectors of real numbers between 0 and 2, corresponding to $16 * 16$ pixels gray-scale images of handwritten digits (examples on Figure 9). They are presented to the network in temporal coding: The higher the numerical value, the darker the pixel color, the earlier the spike firing of the corresponding input neuron, inside a time window of $T = 20$ ms. Hence, the significant part of the pattern (i.e. the localization of the black pixels) is presented first to the network, which is an advantage of temporal processing compared to usual methods that scan the image matrix of pixels line after line.

Fig. 9. USPS patterns examples: 1, 5, 8, 9 digits.

For this task, the neuron model constants and network hyper-parameters have been set as follows:

```
Network architecture:
K = 256 input neurons
M = 100 neurons in the reservoir

Connectivity parameters:
P_in = 0.01 connectivity probability from input neurons toward the reservoir

All other parameters are the same as parameters of Section 4.1.
```

[5] http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html

4.2.1 Two-Class Setting

We first tested the model capacity to discriminate two arbitrarily chosen USPS classes, where each class corresponds to a specific digit. Thus we used a slightly modified version of the stimulation protocol. Indeed, instead of presenting the first class alternating with the second, all training patterns of the two classes were presented in a random order. Several epochs [6] are iterated. Finally, in order to allow error evaluation, all learning mechanisms are stopped and an epoch with training patterns is conducted, followed by a generalization epoch with testing patterns, never presented so far. Performance in generalization dramatically depends on the two classes chosen for the simulation. Left side of Tables 1 and 2 show the results for two representative sets of classes, with a reservoir of 100 neurons. Few cases show very poor success rate. For instance 5 versus 8 (Table 2) yields an error rate close to 12% in generalization. To improve on these cases, we increased the size of the reservoir to 2000 neurons. This slightly improved the rates as shown on the right side of Tables 1 and 2. In particular, 5 versus 8 reaches 4.6% error rate, but the rate of rejected patterns (see Section 4.1.2 for definition) also increased.

	Training	Generalization	Training	Generalization
Error rate	0.42%	2.72%	0.36%	1.81%
Success rate	99.2%	96.8%	99.4%	97.3%
Rejection rate	0.36%	0.45%	0.24%	0.91%

Table 1
Best case of two-class discrimination, on the USPS digits: Rates for classes 1 versus 9 with 100 neurons (left) and 2000 neurons (right) in the reservoir.

	Training	Generalization	Training	Generalization
Error rate	10.7%	12.3%	3.10%	4.60%
Success rate	85.4%	80.7%	90.4%	84.4%
Rejection rate	3.92%	7.06%	6.47%	11.0%

Table 2
Worst case of two-class discrimination, on the USPS digits: Rates for classes 5 versus 8 with 100 neurons (left) and 2000 neurons (right) in the reservoir.

As the dimension of a USPS pattern is about 13 times higher than a pattern from Izhikevich's experiments, we expected more difficulties in order to reach reasonable performance without making any change on the hyper-parameters. Although this is only a two-class classification task, the fact that we already obtain competitive error rates using a reservoir with only 100 neurons is an exciting result, considering the dimension of the patterns and the notoriously "difficult" test set. Those results could be slightly improved with a reservoir of 2000 neurons, which is a common size in reservoir computing literature.

[6] An epoch corresponds to one presentation of all training examples (i.e. several hundreds of patterns, the exact number depending on the digits to be classified).

4.2.2 Multi-Class Setting

A few simulations have been performed on the whole 10 classes of the USPS dataset. Several experimental tunings of the hyper-parameters over the training set led to the following values :

> Network architecture:
> $K = 256$ input neurons
> $M = 2000$ neurons in the reservoir
> 10 readout neurons, instead of 2
>
> Spiking neuron model:
> $\tau_m = 20$ ms for time constant of the membrane potential exponential decrease
>
> Connectivity parameters:
> $P_{in} = 0.01$ connectivity probability from input neurons toward the reservoir
> $P_{rsv} = 0.0145$ connectivity probability inside the reservoir
> $w_{OUT} = 0.02$, fixed value of weights from reservoir to output neurons
> $d_{max} = 100$, maximum delay value, only toward the readouts
>
> All other parameters are the same as parameters of Section 4.1.

We let the simulation go through 20 training epochs before evaluating the rates on the *train* and *test* sets. We obtain an error rate of 8.8% on the training set that jumps to 13.5% on the testing set. Although the performance is not yet competitive with the best well-tuned machine learning approaches that nearly reach 2% in test error (see [20] for a review of performance on the multi-class USPS dataset), the multi-timescale learning rule proves to behave correctly on real-world data. We emphasize that our error rates have been reached after few tunings w.r.t. the size and complexity of the database. First, increasing the size of the reservoir from 100 to 2000 neurons yielded improved performance. Note that setting τ_m (see Section 2.1) to a larger value (i.e. 20, instead of 3), in the readout neurons only, induces a slower decrease of their membrane potential. The neurons are thus tuned to behave as integrators [33].

	Training	Generalization
Error rate	8.87%	13.6%
Success rate	85.0%	79.1%
Rejection rate	6.13%	7.32%

Table 3
Rates for all 10 classes of USPS dataset

Another key point was to set a wider range of possible delay values for the readout connections. Thus, allowing the connections to be set in $[1, 100]$ instead of $[1, 20]$ also improved the success rate, and reduced the rejection rate. This let us think that the discretization of the delays is too low to avoid coincidence of readout spikes. In order to circumvent this effect, a test has been performed with a different time step: 0.1 ms instead of 1 ms, on a reservoir of 200 neurons. The result has been a 7% increase of the generalization success rate, mainly coming from a decrease (-5.5%) of rejected patterns.

Although the multi-class case needs further investigation, we consider these preliminary results and observations as very encouraging. Hyper-parameters (mainly the reservoir size and the connectivity probabilities) have to be tuned. Hence their interactions with the neuron model constants have to be controlled in order to keep a convenient level of activity inside the reservoir. Understanding more deeply the activity and the effects of modifying connectivity in the reservoir will hopefully help to improve the classification performance. Nevertheless, the concept is validated, as confirmed by the analysis of the model behavior presented in the next two sections.

5 Polychronization

5.1 Cell Assemblies and Synchrony

A cell assembly can be defined as a group of neurons with strong mutual excitatory connections. Since a cell assembly tends to be activated as a whole once a subset of its cells are stimulated, it can be considered as an operational unit in the brain. Inherited from Hebb, current thoughts about cell assemblies are that they could play a role of "grandmother neural groups" as basis of memory encoding, instead of the old debated notion of "grandmother cell", and that material entities (e.g. a book, a cup, a dog) and, even more, mental entities (e.g. ideas or concepts) could be represented by different cell assemblies. However, although reproducible spike-timing patterns have been observed in many physiological experiments, the way these spike-timing patterns, at the millisecond scale are related to high-level cognitive processes is still an open question.

Deep attention has been paid to synchronization of firing times for subsets of neurons inside a network. The notion of synfire chain [2,11], a pool of neurons firing synchronously, can be described as follows: If several neurons have a common post-synaptic neuron N_j and if they fire synchronously then their firing will superimpose in order to trigger N_j. However, the argument falls down if the axonal transmission delays are to be considered, since the incoming synapses of N_j have no reason to share a common delay value. Synchronization appears to be a too restrictive notion when it comes to grasp the full power of cell assemblies processing. This point has been highlighted by Izhikevich [14] who proposes the notion of polychronization.

5.2 Polychronous Groups

Polychronization is the ability of an SNN to exhibit reproducible time-locked but not synchronous firing patterns with millisecond precision, thus giving a new light to the notion of cell assembly. Based on the connectivity between neurons, a *polychronous group* (PG) is a possible stereotypical time-locked firing pattern. For example, in

Figure 10, if we consider a delay of 15 ms from neuron N_1 to neuron N_2, and a delay of 8 ms from neuron N_3 to neuron N_2, then neuron N_1 emitting a spike at time t and neuron N_3 emitting at time $t+7$ will trigger a spike firing by neuron N_2 at time $t+15$ (supposing two coincident incoming spikes are enough to make a neuron fire). Since neurons of a polychronous group have matching axonal conduction delays, the group can be the basis of a reproducible spike-timing pattern: Firing of the first few neurons with the right timing is enough to activate most of the group (with a tolerance of 1 ms jitter on spike-timing).

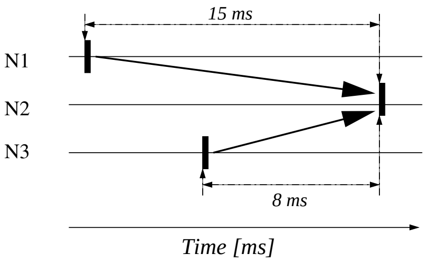
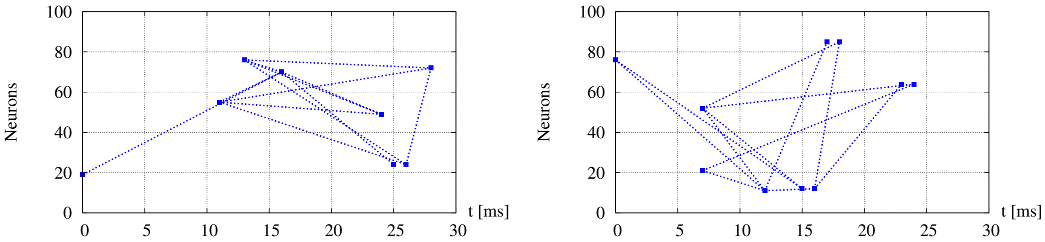


Fig. 10. Example of two triggering neurons giving rise to a third one firing.

Since any neuron can be activated within several PGs, at different times (e.g. neuron 76 in Figure 11), the number of coexisting PGs in a network can be much greater than its number of neurons, thus opening possibility of huge memory capacity. All the potential PGs in a reservoir network of M neurons, depending on its topology and the values of the internal transmission delays (that are kept fixed), can be enumerated using a greedy algorithm of complexity $O(M^{2+F})$, where F is the number of triggering neurons to be considered. In the reservoir used for experiments on Izhikevich's patterns (Section 4.1), we have referenced all the possible polychronous groups inherent in the network topology, with $F = 3$ triggering neurons (see Figure 11 for examples). We have detected 104 potentially activatable PGs in a network of $M = 100$ neurons. In similar conditions, the number of PGs already overcomes 3000 in a network of $M = 200$ neurons.



Triggering neurons: 19,55,76, with timing pattern (0,11,13)

Triggering neurons: 21,52,76, with timing pattern (7,7,0)

Fig. 11. Examples of polychronous groups: PG_50 and PG_51. Starting from three initial triggering neurons, further neurons of the group can be activated, in chain, with respect to the spike-timing patterns represented on the diagrams.

Our model proposes a way to confirm the link between an input presentation and the activation of persistent spike-timing patterns inside the reservoir, and the way we take advantage of polychronous groups for supervising the readout adaptation is explained in the next section.

6 Reservoir Internal Dynamics

Since the number of polychronous groups increases very rapidly when the reservoir size grows (cf. Section 5.2), the dynamical behavior of the network has been deeply examined only for the two-class experiments on Izhikevich's patterns, where the network size remains small (104 PGs only). All along the initialization, learning and generalization phases, the reservoir internal dynamics has been analyzed in terms of actually activated polychronous groups. Figure 12 presents the evolution of polychronous groups activation in experiments. The evolution of activated PGs is entirely governed by STDP, the only adaptation process acting inside the reservoir network.

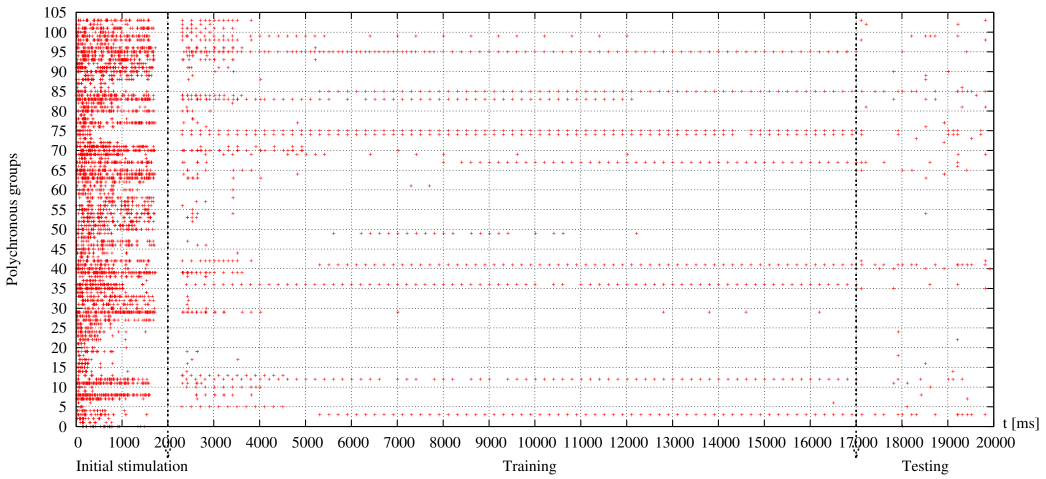


Fig. 12. Evolution of polychronous groups activation during the initialization, learning and generalization phases of experiments reported in Section 4.1, with alternate class input presentations. Note that, in this figure, the y-axis represents the PG indices and no longer the neuron indices, as was the case on spike raster plots.

We observe that many PGs are frequently activated during the initial random stimulation that generates a strong disordered activity in the internal network (before 2000 ms). At the beginning of the learning phase (which goes from 2000 ms to 17000 ms), many groups are activated, and then, roughly after 5000 ms, the activation landscape becomes very stable. As anticipated, only a few specific polychronous groups continue to be busy. Small subsets of PGs can be associated to each class: Groups 3, 41, 74, 75 and 83, switching to 85, fire for class 1, whereas groups 12, 36, 95, sometimes 99, and 49, switching to 67, fire for class 2. During the generalization phase (after 17000 ms), the main and most frequently activated groups are those identified during the learning phase. This observation supports the hypothesis that polychronous groups have become representative of the class encoding realized by the multi-scale learning rule.

Several interesting observations can be reported. As noticed by Izhikevich, there exist groups that start to be activated only after a large number of repeated stimulations (e.g. 41, 49, 67 and 85), whereas some other groups stop their activation after a while (e.g. 5 and some others [active until 4000/5000 ms only], 49, 70, 83 and 99 [later]). We can also observe that PGs specialize for one particular class (later than 8000 ms) instead of answering for both of them, as they did first (mainly from 2000 to 5000 ms).

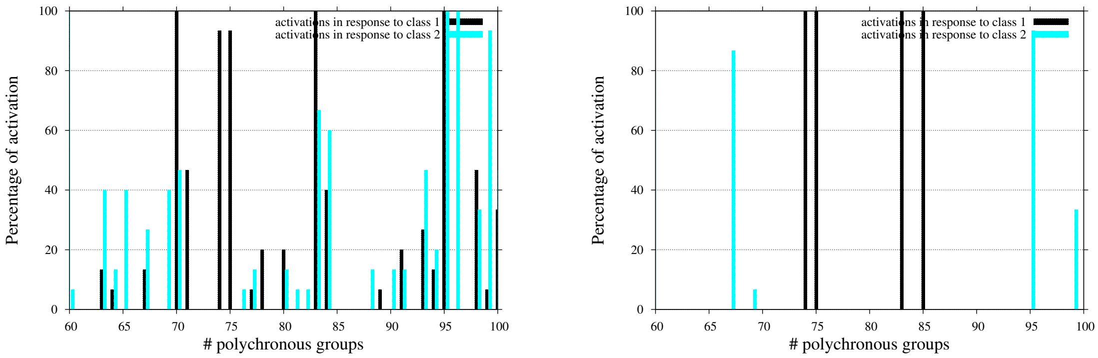


Fig. 13. Activation ratio from 2000 to 5000ms, and then from 8000 to 11000ms.

An histogram representation of a subset of the 104 PGs (Figure 13) better points out the latter phenomenon. A very interesting case is the polychronous group number 95 which is first activated by both example patterns, and then (around time 7500 ms) stops responding for class 1, thus specializing its activity for class 2. Such phenomenon validates that synaptic plasticity provides the network with valuable adaptability and highlights the importance of combining STDP with delay learning.

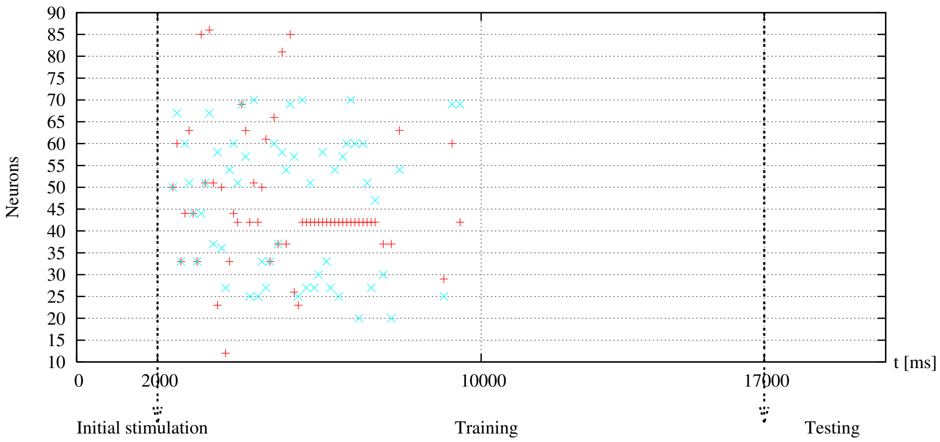


Fig. 14. The 80 excitatory neurons of the reservoir have been marked by a red (black) "+" for class 1 or a blue (light grey) "x" for class 2 each time they play the role of pre-synaptic neuron of a triggering connection producing a readout delay change.

The influence of active PGs on the learning process can also be exhibited. We have recorded (Figure 14) the indices of the pre-synaptic neurons responsible for the application of the output delay update rule, at each iteration where the example pattern was not yet well classified (cf. Algorithm 1, Section 3.2). For instance, neuron #42, which is repeatedly responsible for delay adaptation, is one of the triggering neurons of the polychronous group number 12, activated for class 2 during the training phase. One will notice that delay adaptation stops before the learning phase is over, which means the learning process is already efficient around 10000 ms. Such a control could be implemented in the proposed algorithms, as a heuristic for a better stopping criterion (rather than the current "*until* a given maximum learning time is over").

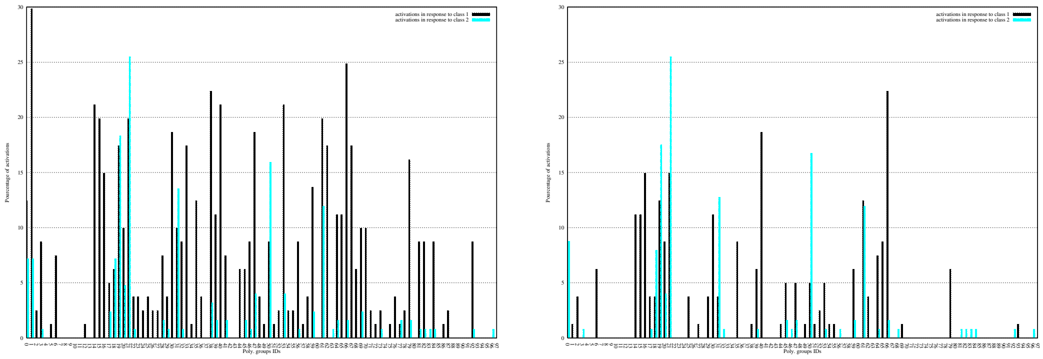


Fig. 15. Activation ratio of PGs in a 100 neurons reservoir, for two-class discrimination of USPS digits 1 versus 9, during the 1^{st} (left) and the 5^{th} (right) epochs.

Figure 15 confirms the selection of active polychronous groups during the learning process, even on complex data. Although the phenomenon is less precise, due to the high variability of USPS patterns inside each class, it still remains observable: After only 5 epochs of the learning phase, the activity of many PGs has vanished and several of them are clearly specialized for one class or the other.

7 Conclusion

We have proposed a new model for Reservoir Computing, based on a multi-timescale learning mechanism for adapting a Spiking Neuron Network to a classification task. The proof of concept is based on the notion of polychronization. Under the effect of synaptic plasticity (STDP), the reservoir network dynamics induces the emergence of a few active polychronous groups specific to the patterns to be discriminated. The delay adaptation mechanism of the readout neurons makes them capture the internal activity so that the target class neuron fires before the other ones, with an enforced time delay margin. Adaptation to the task at hand is based on biological inspiration. The delay learning rule is computationally easy to implement and gives a way to supervise the overall process. Performance on two-class discrimination tasks is reasonably good, even with a small size of reservoir network, on notoriously difficult patterns. Deeper investigation on the interactions between the hyper-parameters would help to improve the performance. A first track consists in running the reservoir simulation with a smaller time step for large and noisy databases.

While the notion of margin is important in modern machine learning literature, it needs to be paired with some form of regularization. We thus intend to also explore ways to implement a regularization process in Reservoir Computing in the near future. Another perspective is to adapt the method to regression tasks or time series prediction in order to stronger exploit the opportunity of temporal processing in the reservoir. In future work we will test our classification task on other reservoir computing models in order to compare the results of our model to those of the literature. We will use the Reservoir Computing Toolbox available at http://www.elis.ugent.be/rct [38].

Acknowledgement

The authors acknowledge David Meunier for precious help and clever advice about the most pertinent way to implement STDP and anonymous reviewers for valuable suggestions.

References

[1] L. Abbott, S. Nelson, Synaptic plasticity: taming the beast, Nature Neuroscience 3 (2000) 1178–1183.

[2] M. Abeles, Corticonics: Neural Circuits of the Cerebral Cortex, Cambridge Press, 1991.

[3] G.-q. Bi, M.-m. Poo, Synaptic modification in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and polysynaptic cell type, J. of Neuroscience 18 (24) (1998) 10464–10472.

[4] S. Bohte, J. Kok, H. La Poutré, Spikeprop: Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (2002) 17–37.

[5] S. Bohte, M. Mozer, Reducing the variability of neural responses: A computational theory of spike-timing-dependent plasticity., Neural Computation 19 (2) (2007) 371–403.

[6] V. Braitenberg, A. Schuz, Anatomy of the cortex: Statistics and geometry, Springer-Verlag, 1991.

[7] N. Butko, J. Triesch, Learning sensory representations with intrinsic plasticity, Neurocomputing 70 (2007) 1130–1138.

[8] G. Chechik, Spike-timing dependent plasticity and relevant mutual information maximization, Neural Computation 15 (7) (2003) 1481–1510.

[9] G. Daoudal, D. Debanne, Long-term plasticity of intrinsic excitability: Learning rules and mechanisms, Learning and Memory 10 (2003) 456–465.

[10] N. Desai, L. Rutherford, G. Turrigiano, Plasticity in the intrinsic excitability of cortical pyramidal neurons, Nature Neuroscience 2 (6) (1999) 515–520.

[11] M. Diesmann, M.-O. Gewaltig, A. Aertsen, Stable propagation of synchronous spiking in cortical neural networks, Nature 402 (1999) 529–533.

[12] W. Gerstner, W. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

[13] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning, Springer-Verlag, 2001.

[14] E. Izhikevich, Polychronization: Computation with spikes, Neural Computation 18 (2) (2006) 245–282.

[15] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, Tech. Rep. TR-GMD-148, German National Research Center for Information Technology (2001).

[16] H. Jaeger, Adaptive nonlinear system identification with Echo State Networks, in: S. Becker, S. Thrun, K. Obermayer (eds.), NIPS*2002, Advances in Neural Information Processing Systems, vol. 15, MIT Press, 2003.

[17] H. Jaeger, W. Maass, J. Principe, Special issue on echo state networks and liquid state machines (editorial), Neural Networks 20 (3) (2007) 287–289.

[18] E. Kandell, J. Schwartz, T. Jessell, Principles of Neural Science, 4th edition, McGraw-Hill, 2000.

[19] R. Kempter, W. Gerstner, J. L. van Hemmen, Hebbian learning and spiking neurons, Physical Review E 59 (4) (1999) 4498–4514.

[20] D. Keysers, R. Paredes, H. Ney, E. Vidal, Combination of tangent vectors and local representations for handwritten digit recognition, in: SPR 2002, International Workshop on Statistical Pattern Recognition, Windsor, Ontario, Canada, 2002.

[21] A. Lazar, G. Pipa, J. Triesch, Fading memory and time series prediction in recurrent networks with different forms of plasticity, Neural Networks 20 (3) (2007) 312–322.

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

[23] W. Maass, On the relevance of time in neural computation and learning, Theoretical Computer Science 261 (2001) 157–178, (extended version of ALT'97, in LNAI 1316:364-384).

[24] W. Maass, T. Natschläger, Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding, Network: Computation in Neural Systems 8 (4) (1997) 355–372.

[25] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Computation 14 (11) (2002) 2531–2560.

[26] W. Maass, M. Schmitt, On the complexity of learning for a spiking neuron, in: COLT'97, Conf. on Computational Learning Theory, ACM Press, 1997.

[27] W. Maass, M. Schmitt, On the complexity of learning for spiking neurons with temporal coding, Information and Computation 153 (1999) 26–46.

[28] H. Markram, J. Lübke, M. Frotscher, B. Sakmann, Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs, Science 275 (1997) 213–215.

[29] D. Meunier, H. Paugam-Moisy, Evolutionary supervision of a dynamical neural network allows learning with on-going weights, in: IJCNN'2005, Int. Joint Conf. on Neural Networks, IEEE–INNS, 2005.

[30] T. Natschläger, B. Ruf, Spatial and temporal pattern analysis via spiking neurons, Network: Comp. Neural Systems 9 (3) (1998) 319–332.

[31] D. Norton, D. Ventura, Preparing more effective liquid state machines using hebbian learning, in: IJCNN'2006, Int. Joint Conf. on Neural Networks, IEEE-INNS, 2006.

[32] T. Nowotny, V. Zhigulin, A. Selverston, H. Abardanel, M. Rabinovich, Enhancement of synchronization in a hibrid neural circuit by Spike-Time-Dependent Plasticity, The Journal of Neuroscience 23 (30) (2003) 9776–9785.

[33] H. Paugam-Moisy, Spiking neuron networks: A survey, IDIAP-RR 11, IDIAP (2006).

[34] H. Paugam-Moisy, R. Martinez, S. Bengio, A supervised learning approach based on STDP and polychronization in spiking neuron networks, Tech. Rep. 54, IDIAP, http://www.idiap.ch/publications/paugam-esann-2007.bib.abs.html (october 2006).

[35] J.-P. Pfister, T. Toyoizumi, D. Barber, W. Gerstner, Optimal spike-timing dependent plasticity for precise action potential firing in supervised learning, Neural Computation 18 (6) (2006) 1318–1348.

[36] J. Rubin, D. Lee, H. Sompolinsky, Equilibrium properties of temporally assymmetric Hebbian plasticity, Physical Review Letters 89 (2) (2001) 364–367.

[37] M. Schmitt, On computing boolean functions by a spiking neuron, Annals of Mathematics and Artificial Intelligence 24 (1998) 181–191.

[38] B. Schrauwen, D. Verstraeten, J. Van Campenhout, An overview of reservoir computing: theory, applications and implementations, in: M. Verleysen (ed.), ESANN'2007, Advances in Computational Intelligence and Learning, 2007.

[39] W. Senn, M. Schneider, B. Ruf, Activity-dependent development of axonal and dendritic delays, or, why synaptic transmission should be unreliable, Neural Computation 14 (2002) 583–619.

[40] J. Sima, J. Sgall, On the nonlearnability of a single spiking neuron, Neural Computation 17 (12) (2005) 2635–2647.

[41] W. Singer, Neural synchrony: A versatile code for the definition of relations?, Neuron 24 (1999) 49–65.

[42] J. Steil, Backpropagation-decorrelation: Online recurrent learning with O(N) complexity, in: IJCNN'2004, Int. Joint Conf. on Neural Networks, IEEE–INNS, 2004.

[43] J. Steil, Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning, Neural Networks 20 (3) (2007) 353–364.

[44] H. Swadlow, Physiological properties of individual cerebral axons studied in vivo for as long as one year, Journal of Neurophysiology 54 (1985) 1346–1362.

[45] H. Swadlow, Monitoring the excitability of neocortical efferent neurons to direct activation by extracellular current pulses, J. Neurophysiology 68 (1992) 605–619.

[46] T. Toyoizumi, J.-P. Pfister, K. Aihara, W. Gerstner, Generalized Bienenstock - Cooper - Munro rule for spiking neurons that maximizes information transmission, Proc. Natl. Acad. Sci. 102 (14) (2005) 5239–5244.

[47] J. Triesch, A gradient rule for the plasticity of a neuron's intrinsic excitability, in: ICANN'05, Int. Conf. on Artificial Neural Networks, 2005.

[48] V. Vapnik, Statistical learning theory, Wiley, 1998.

[49] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Networks 20 (3) (2007) 391–403.

[50] M. Wardermann, J. Steil, Intrinsic plasticity for reservoir learning algorithms, in: M. Verleysen (ed.), ESANN'2007, Advances in Computational Intelligence and Learning, 2007.

[51] M. Woodin, K. Ganguly, M. Poo, Coincident pre- and postsynaptic activity modifies gabaergic synapses by postsynaptic changes in CL-transporter activity, Neuron 39 (5) (2003) 807–820.