Training Spiking Neural Networks to Associate Spatio-temporal Input-Output Spike Patterns

Ammar Mohemmed[a], Stefan Schliebs[a], Satoshi Matsuda[c], Nikola Kasabov[a,b]

[a]*Knowledge Engineering and Discovery Research Institute,*
350 Queen Street, Auckland 1010, New Zealand
{amohemme,sschlieb,nkasabov}@aut.ac.nz
www.kedri.info
[b] *Institute for Neuroinformatics, ETH and University of Zurich*
[c]*Department of Mathematical Information Engineering,*
Nihon University, Japan
matsuda.satoshi@nihon-u.ac.jp

Abstract

In a previous work (Mohemmed et al., Method for training a spiking neuron to associate input-output spike trains) [1] we have proposed a supervised learning algorithm based on temporal coding to train a spiking neuron to associate input spatiotemporal spike patterns to desired output spike patterns. The algorithm is based on the conversion of spike trains into analogue signals and the application of the Widrow-Hoff learning rule. In this article we present a mathematical formulation of the proposed learning rule. Furthermore, we extend the application of the algorithm to train a SNN consisting of multiple spiking neurons to perform spatiotemporal pattern classification and we show that the accuracy of classification is improved significantly over a single spiking neuron. We also investigate a number of possibilities to map the temporal output of the trained spiking neuron into a class label. Potential applications for motor control in neuro-rehabilitation and neuro-prosthetics are discussed as a future work.

Keywords: Spiking Neural Networks, Supervised learning, Spatio-temporal control, Temporal coding

1. Introduction

The perfection exhibited by living entities in carrying out their daily natural activities is inspiring researchers to adopt their behavior, deep to the cell level, as a model to solve computational tasks that are considered complex for machines to solve. The study of Spiking Neural Networks (SNN) [2, 3, 4, 5] represents a significant step in the path of learning from the brain. SNN is closer to the real operational model of the brain than conventional neural networks. This closeness is asserted in the use of spikes as a form of communication between the neural nodes similar to the brain. The shape of the spike seems less relevant and has no importance in representing the information, instead the time of spiking carries the information. How information is encoded in the spike timing is a debatable issue as many theories exist. Traditionally, the commonly used neural code in SNN is rate coding in which the information is encoded in the number of spikes over a small time window. Alternatively, the temporal coding encodes the information in the exact timing of the spikes. Information representation has an important role in simplifying and speeding the computation to achieve good results. In [6] it was argued that the recognition of patterns such as colors, visual patterns, odours and sound quality are solved rapidly in neurobiology using temporal coding and could not be solved using rate-based neural models. Furthermore, temporal coding is supported by evidences observed in different types of biological neurons, see [7] for a survey.

The other issue establishing the biological plausibility of SNN is the learning paradigm referred to as Spike Time Dependent Plasticity (STDP) [8, 9, 2, 10]. The STDP is an unsupervised learning process that adjusts the synaptic weights based on the time correlation between the incoming spike (presynaptic spike) and the emitted spike of the neuron (postsynaptic spike). In [11] it was shown that STDP enables a neuron to perform a complex recognition task: to localize a repeating spatiotemporal spike pattern embedded in equally dense distractor spike trains. In [12], an unsupervised learning algorithm based on STDP and Winner-Take-All (WTA) paradigm is proposed for pattern recognition.

However, for specific task oriented engineering applications, supervised learning (training) or a combined unsupervised-supervised, might be more favourable over unsupervised learning. Supervised learning, commonly in the form of error back propagation [13], is widely used in training conventional neural networks to perform pattern recognition. Due to the nature of spike-based communication and the complexity of SNN (which requires tuning big number of parameters), no efficient supervised learning techniques for SNN have existed until recently.

One of the first supervised learning methods for SNN is SpikeProb [14]. This uses a gradient descent approach that adjusts the synaptic weights in order to emit a single spike at a specified time. The timing of the output spike encodes specific information, *e.g.* the class label of the presented input sample. However, SpikeProp cannot train SNN to emit a desired spike train consisting of more than one spike.

An interesting learning rule for spatiotemporal pattern recognition has been suggested in [12]. The so-called Tempotron enables a neuron to learn whether to fire or not to fire in response to a specific input stimulus. Consequently, the method allows the processing of binary classification problems. However, the neuron is not intended to learn a precise target output spike train, but instead whether to spike or not to spike in response to an input stimulus.

A Hebbian based supervised learning algorithm called Remote Supervised Method (ReSuMe) was proposed in [15] and further studied in [16, 17]. ReSuMe, similar to STDP, is based on a learning window concept. Using a teaching signal a specific desired output is imposed on the output neuron. With this method, a neuron can produce a spike train precisely matching a desired spike train. It was shown that in combination with the Liquid State Machine (LSM) [18], the algorithm is efficient for random mapping from any input spike train to any output spike train or multiple spike trains. The algorithm was mainly designed and applied for neuroprostheses control [19].

Recently, a method called Chronotron was proposed [20]. Two versions of learning rules are described therein; E-learning and I-learning. E-learning is based on minimizing the error between the desired spike pattern and the actual one. The error is measured using the Victor-Purpura spike distance metric [21]. This metric produces discontinuities in the error landscape that must be overcome through approximation. E-Learning surpasses ReSuMe in terms of the number of spike patterns that can be memorized and classified. The other version, I-Learning, is biologically more plausible but less efficient.

In [22] the authors proposed a supervised learning paradigm for SNN based on Particle Swarm Optimization (PSO). PSO optimizes, according to a fitness function, the parameters of the dynamic synapses [23] which connect the layers of the network. The fitness function measures the similarity between the actual output spike train and the target spike train. However, PSO becomes less efficient at finding good solutions when the number of variables (*i.e.* the parameters of the synapses) increases, limiting its applicability for large networks, especially when the input stimulus is a spatiotemporal spike pattern consisting of many spike trains. To overcome this difficulty, the authors proposed in [1] a simple method to train a neuron to map (associate) an input spatiotemporal spike pattern to a desired spike train pattern. The method is based on the Widrow-Hoff (or Delta) learning rule [24] commonly used in traditional neural networks. The Delta rule adjusts the weight of a synapse by scaling the error signal, *i.e.* the difference between the teacher signal and the actual signal, by the value of the input at that synapse. The Delta rule is inapplicable to SNN because spikes, unlike real-values signals, cannot be subtracted or multiplied directly. In the mentioned proposed learning rule, spike trains are converted into continuous signals by convolution with a kernel function. The Delta rule can then be applied directly to adjust the synaptic weight for training purposes. We refer to a spiking neuron trained via this method by SPAN (Spike Pattern Association Neuron) since the neuron is intended primarily for input/output spike pattern association. SPAN was evaluated to be efficient in a synthetic spatiotemporal classification problem [1].

In this article, a mathematical formulation of SPAN learning rule is provided. Furthermore, instead of a single SPAN to perform spatiotemporal classification, we train multiple SPANs in a single layer network to perform the classification task and compare the accuracy with that of a single SPAN. Because SPAN is based on temporal coding, we describe and test different ways to transform the output spike pattern into a class label.

In the next section the learning rule is described and derived mathematically. In section 3 we discuss the multiple SPAN architecture. In section 4, the details of the simulation experiments and results using multiple SPANs are given. Section 5 concludes the paper and highlights future research and applications.

2. The SPAN learning rule

Similar to other supervised training algorithms, the synaptic weights of the network are adjusted iteratively to impose a desired input/output spike pattern association to the SNN. To derive the learning rule, we begin with Widrow-Hoff rule as follows. For a synapse i, the weight change Δw_i is defined as:

$$\Delta w_i = \lambda x_i \, (y_d - y_{out}) = \lambda x_i \Delta_i \tag{1}$$

where $\lambda \in \mathbb{R}$ is a real-valued positive learning rate, x_i is the input transferred through synapse i, and y_d and y_{out} refer to the desired and the actual neural output, respectively. Note that $\Delta_i = y_d - y_{out}$ is the difference or error between the desired and the actual output of the neuron.

This rule was introduced for conventional neural networks where the input and output are real-valued signals. In SNN however, trains of spikes are passed between neurons rendering the Widrow-Hoff rule incompatible for SNN. More specifically, if x_i, y_d and y_{out} are considered as spike trains $s(t)$ defined by

$$s(t) = \sum_f \delta(t - t^f) \tag{2}$$

where t^f is the firing time of a spike and $\delta(\cdot)$ is the Dirac delta function $\delta(x) = 1$ if $x = 0$ and 0 otherwise, then the difference between two spike trains y_d and y_{out} does not define a suitable error landscape which can be minimized by a gradient descent method.

Here, we address this issue by proposing the following idea. In order to define the difference between spike trains, we convolve each spike sequence with a kernel function $\kappa(t)$. This is similar to the binless distance metric used to compare spike trains [25]. We define

$$\tilde{x}_i(t) = \sum_{t_i^f \in F_{\text{in}}} \kappa(t - t_i^f) \tag{3}$$

$$\tilde{y}_d(t) = \sum_{t_d^g \in F_{\text{d}}} \kappa(t - t_d^g) \tag{4}$$

$$\tilde{y}_{out}(t) = \sum_{t_{out}^h \in F_{\text{out}}} \kappa(t - t_{out}^h) \tag{5}$$

with F_{in}, F_{d} and F_{out} being the input, the desired and the actual output set of spike trains, respectively. Substituting x_i, y_d and y_{out} with the kernelized spike trains $\tilde{x}_i(t)$, $\tilde{y}_d(t)$ and $\tilde{y}_{out}(t)$, a new learning rule for a spiking neuron is obtained:

$$\Delta w_i(t) = \lambda \tilde{x}_i(t) \, (\tilde{y}_d(t) - \tilde{y}_{out}(t)) \tag{6}$$

This equation formulates a real-time learning rule such that the synaptic weights change over time. By integrating Eq. 6, we derive the batch version of the learning rule which is under scrutiny in this paper:

$$\Delta w_i = \lambda \int_0^\infty \tilde{x}_i(t) \, (\tilde{y}_d(t) - \tilde{y}_{out}(t)) \, \mathrm{dt} \tag{7}$$

A variety of kernel functions $\kappa(t)$ exist such as linear, (double) exponential, alpha and Gaussian kernels. In this study, we use an α-kernel, $\alpha(t) = e \, \tau^{-1} \, t \, e^{-t/\tau} H(t)$, although many other kernels could have been chosen. A convolved spike train $\tilde{s}(t)$ is then given as:

$$\begin{aligned} \tilde{s}(t) &= \sum_{t^f} \kappa(t - t^f) \\ &= \sum_{t^f} e \, \tau^{-1} \, (t - t^f) \, e^{-(t-t^f)/\tau} H(t - t^f) \end{aligned} \tag{8}$$

where $H(t)$ refers to the Heaviside function and $\tau \in \mathbb{R}$ is a real-valued time constant. Using this kernel function, Eq. 6 is rewritten as follows:

$$\begin{aligned} &\Delta w_i(t) = \\ &\lambda \left(\frac{e}{2}\right)^2 \Bigg[\sum_g \sum_f H(t - \max\{t_i^f, t_d^g\})(t - t_d^g)(t - t_i^f) e^{-\frac{2t - t_i^f - t_d^g}{\tau}} \\ &- \sum_h \sum_f H(t - \max\{t_i^f, t_{out}^h\})(t - t_{out}^h)(t - t_i^f) e^{-\frac{2t - t_i^f - t_{out}^h}{\tau}} \Bigg] \end{aligned} \tag{9}$$

Now, we can integrate Eq. 7:

$$\begin{aligned} \Delta w_i &= \lambda \int_0^\infty \Delta w_i(t) \, \mathrm{dt} \\ &= \lambda \left(\frac{e}{2}\right)^2 \Bigg[\sum_g \sum_f (|t_i^f - t_d^g| + \tau) e^{-\frac{|t_i^f - t_d^g|}{\tau}} \\ &\quad - \sum_h \sum_f (|t_i^f - t_{out}^h| + \tau) e^{-\frac{|t_i^f - t_{out}^h|}{\tau}} \Bigg] \end{aligned} \tag{10}$$

Eq. 10 defines how the weight of a synapse changes after presenting a training sample in each epoch.

In each iteration (or epoch), all input patterns are presented sequentially to the system. For each pattern the Δw_i are computed and accumulated. After the presentation of all patterns, the weights are updated to $w_i(e + 1) = w_i(e) + \Delta w_i(e)$, where e is the current epoch of the learning process.

Fig. 1 illustrates the functioning of the learning method. An output neuron is connected to three input neurons through three excitatory synapses with randomly initialized weights. For simplicity, each input

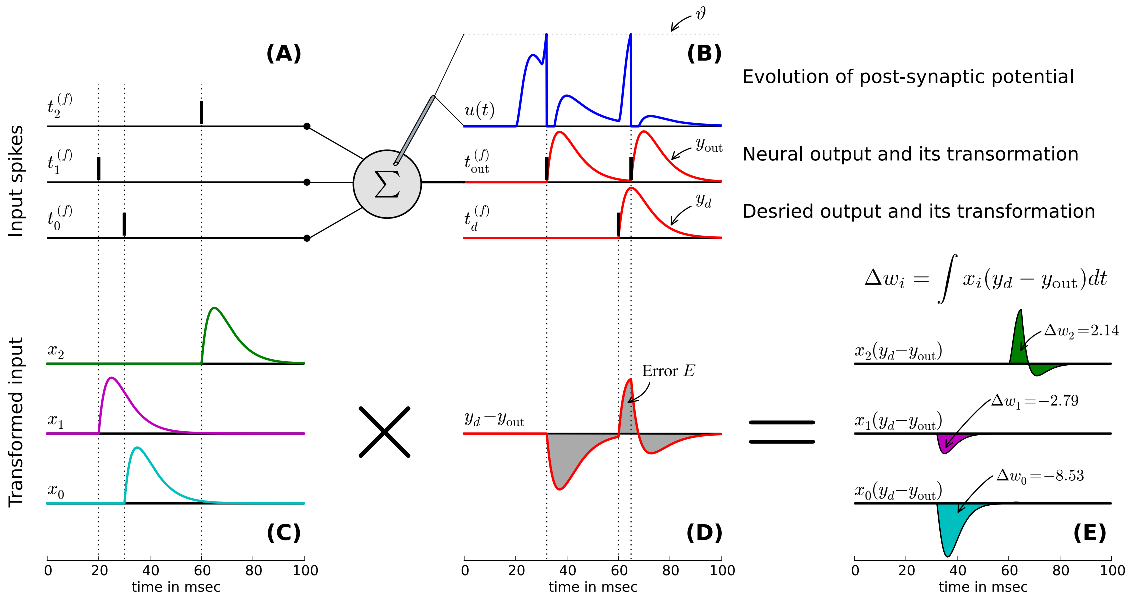


Figure 1: Illustration of the proposed learning rule SPAN. See Section 2 for detailed explanations of the figure.

sequence consists of a single spike only. However, the learning method can also deal with more than one spike per input neuron. The inputs $t_i^{(f)}$ are visualized in Fig. 1A. In this example, we intend to train the output neuron to emit a single spike at a pre-defined time $t_d^{(0)}$.

Assume that, as shown in Fig. 1B, the presented stimulus excites the output neuron resulting in the generation of two output spikes at times $t_{\text{out}}^{(0)}$ and $t_{\text{out}}^{(1)}$, respectively, neither of them being the desired spike time $t_d^{(0)}$. The evolution of the measured membrane potential $u(t)$ in the output neuron is shown in the upper section of Fig. 1B above the actual and the desired spike trains.

The lower parts of Fig. 1 (C,D,E) graphically illustrate Eq. 7. The input, actual and desired spikes trains are convolved with the α-kernel as defined in Eq. 8 (Fig. 1B and C). We define the area under the curve of the difference $y_d(t) - y_{\text{out}}(t)$ as an error between actual and desired output:

$$E = \int |y_d(t) - y_{\text{out}}(t)|\, dt \tag{11}$$

Although this error is not used in computing the weight updates Δw_i, this metric is an informative measure of the achieved training status of the output neuron. Fig. 1E shows the weight updates Δw_i. We especially note the large decrease of weight w_0. The spike train $t_0^{(0)}$ of the first input neuron causes an undesired spike at $t_{\text{out}}^{(0)}$ and lowering the corresponding synaptic efficacy potentially suppresses this behavior. On the other hand, the synaptic weight w_2 is increased promoting the triggering of spike $t_{\text{out}}^{(1)}$ at an earlier time.

We note that, unlike related methods such as ReSuMe [15], the defined learning rule employs no learning windows, rendering the method easy to comprehend and to implement.

We demonstrate the spike pattern association function of SPAN in the following task. The task is to learn a mapping from a random input spike pattern to specific target output spike train. This target train consists of five spikes occurring at different times, *i.e.* , $t_d = \{33, 66, 99, 132, 165\}$ms. Initially, the synaptic weights are randomly generated uniformly in the range $(0, 25\text{pA})$; the parameters of the simulation are given in section 4.1. In 100 epochs, we allow the output neuron to adjust its connection weights in order to produce the desired output spike train. The experiment is repeated for 100 runs, each of them initialized with different random weights to guarantee statistical significance.

In Fig. 2, the experimental setup of a typical run is illustrated. The left side of the diagram shows the SPAN network architecture. The right side shows the desired target spike train (top) along with the produced spike trains by the output neuron over a number of learning epochs (bottom). We note that the output spike trains in early epochs are very different from the desired target spike sequence. In later epochs the output spikes converge towards the desired sequence. Consequently, the error as defined in Eq. 11 decreases in succeeding epochs (right part of Fig. 2). We note that the neuron

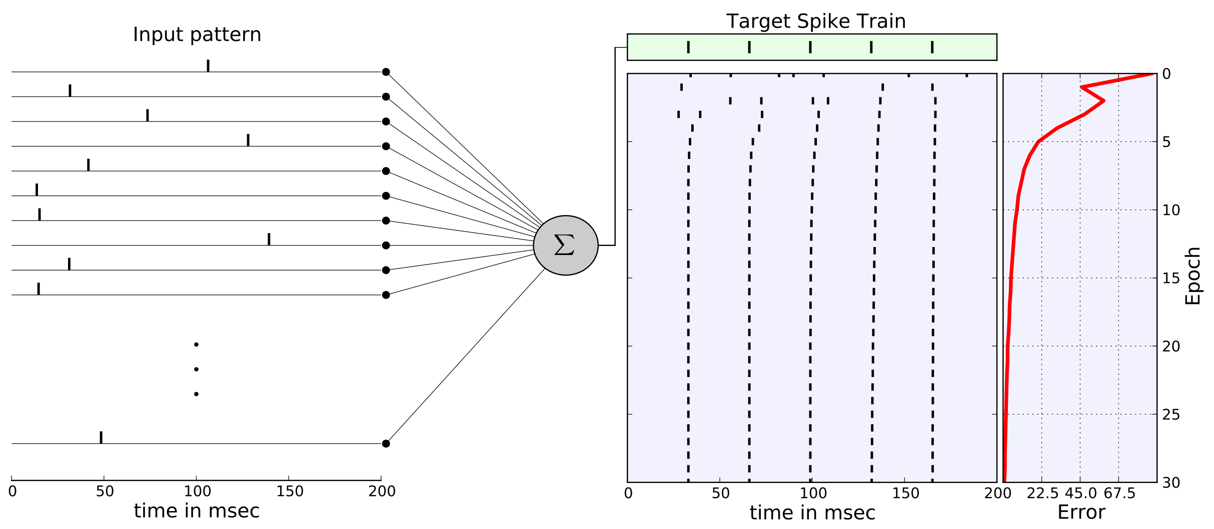


Figure 2: A single output neuron is trained to respond with a temporally precise output spike train to a specific spatiotemporal input. The organization of the figure is inspired by [20].

reproduces the desired spike output pattern very precisely in less than 30 learning epochs. More complex target trains, with more spikes or longer time, have also been tested with considerable success (results are not shown). However, it is necessary that enough spikes are input to stimulate the neuron, otherwise the neuron will not readily reproduce the target train in cases of few synapses. This situation has been previously noted in [17].

3. Training Multiple SPANs

With temporal coding, the label of a spike pattern is determined by not only the occurrence or nonoccurence of a spike, but also by the precise timing of the spike, which can be anywhere within the time period of the simulation. This introduces flexibility and redundancy in deciding the class label from the spike output of the neuron. The question arises as to how many spike patterns the neuron can be trained to remember and consequently to recognize. The answer, discussed in [12], is that the memory capacity of the neuron depends on the synapse number, quantified by a measurement called the load factor, (defined as the ratio of the number of input patterns p the neuron can classify correctly to the number of synapses n, *i.e.* $\frac{p}{n}$).

The memory capacity of SPAN has been studied in [27]. The procedure followed to measure the memory capacity, is similar to that presented in [12], is to generate a number of random spike patterns and assign them *randomly* to several classes (five in this case). The task is to train the neuron to classify the patterns correctly, see [27] for more details. For example, the neuron is able to learn and recognize, with high accuracy, a total of 15 patterns with 200 synapses and 35 patterns when assigned 600 synapses. Therefore, the memory capacity of the neuron is expected to increase as the number of synapses increases although more synapses means patterns with more spike trains. Memory should also increase if the class patterns are associated to each other rather than being completely random.

In [1] we have used a single SPAN in a spatiotemporal spike pattern classification problem in which the neuron is trained to recognize five classes by firing at five time instances assigned to identify the classes.

Here, we investigate the spatiotemporal classification based on different criterion to improve the classification accuracy. The criterion is based on two key points: first, instead of a single SPAN trained to classify all classes as in our previous study [1], several SPANs are trained to perform the classification cooperatively, each assigned to recognize a single class only. Fig. 3 depicts the multiple SPAN architecture with five neurons. The hypothesis is that a better accuracy will be achieved, also testing the potential scalability of the method for large scale applications.

In the training phase, a neuron learns to fire at a specific time when the patterns of the respective class are supplied to its input synapses. For example, for the five class problem there are five neurons, the first neuron is

Table 1: Tabular description of the experimental setup as suggested in [26].

Model Summery	
Neural model	Leaky integrate-and-fire
Synaptic model	α shaped synaptic currents
Input	Random input
Connectivity	All input neurons are connected to a single output neuron
Neural Model	
Type	Leaky integrate-and-fire (LIF) neuron
Description	Dynamics of membrane potential $u(t)$:
	• Spike times: $t^{(f)} : u(t^{(f)}) = \vartheta$
	• Sub-threshold dynamics: $\tau_m \frac{du}{dt} = -u(t) + R\, I^{\text{syn}}(t)$
	• Reset & refractoriness: $u(t) = u_r \forall f : t \in (t^{(f)}, t^{(f)} + \tau_{\text{ref}})$
	• exact integration with temporal resolution dt
Parameters	Membrane time constant $\tau_m = 10\text{ms}$
	Membrane resistance $R = 333.33\text{M}\Omega$
	Spike threshold $\vartheta = 20\text{mV}$, reset potential $u_r = 0\text{mV}$
	Refractory period $\tau_{\text{ref}} = 3\text{ms}$
	Time resolution $dt = 0.1\text{ms}$, simulation time $T = 200\text{ms}$
Synaptic Model	
Type	Current synapses with $\alpha-$function shaped post-synaptic currents (PSCs)
Description	Synaptic input current $I^{\text{syn}}(t) = \sum w \sum_f \alpha(t - t^{(f)})$
	$\alpha(t) = \begin{cases} e\,\tau_s^{-1}\, t\, e^{-t/\tau_s}, & \text{if} t > 0 \\ 0, & \text{otherwise} \end{cases}$
Parameters	Synaptic weight $w \in \mathbb{R}$, uniformly randomly initialized in $[0, 25]$
	Synaptic time constant $\tau_s = 5\text{ms}$
Input Model	
Type	Random input
Details	Population of 200 input neurons each firing a single spike at a randomly chosen time in the period $(0, T)$

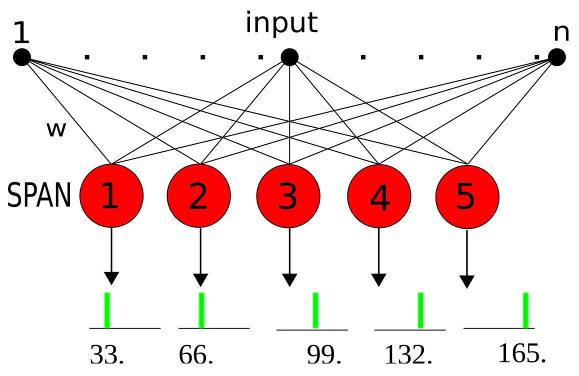


Figure 3: Architecture of the multiple SPANs network. Each neuron is trained to recognize one class by firing at specific time instance.

assigned to class one and spikes at 33ms, the second neuron spikes at 66ms to identify the second class, the third neuron spikes at 99ms to identify the third class and so on. We allow the synaptic weights of the neuron to be adjusted by its assigned class patterns, *i.e.* only the patterns of class 1 are used to adjust the weights of neuron 1, the patterns of class 2 are used to adjust the weights of neuron 2 etcetera. In this way, the neuron, after training, will be selective to the patterns of its assigned class, (equivalent to training each neuron independently of other neurons).

This mechanism is feasible because the training is based on temporal coding and the synaptic weights are adjusted based on the temporal structure of the input patterns. Considering spike patterns, this temporal structure is likely to be unique to every input class. The neuron, after training will be selective to respond properly to this structure. When the neuron is stimulated by other patterns from different classes that have different temporal structure, it will fire in a different way, for example at different time instance or to fire more spikes or even not to fire. A similar mechanism was used in [17] to train the readout neurons of a reservoir (LSM) structure in a spike pattern classification task that originally proposed in [28, 18]. The task was to train the readout neurons to recognize segments of a single long spike train. Each segment represents a specific spike category that the readout neurons are trained to recognize by firing certain spike patterns.

However, it is noted that there are few factors might affect the recognition performance. For example, the different classes might have a very similar temporal structure or quite complex (dense) spike patterns that is hard for the neurons to capture and learn. Therefore, the performance highly depends on the application and the data.

The second key point regarding classification using multiple neurons is how to decide the class label of a pattern given the actual spike output of the neurons. In our previous paper [1], if the neuron fired a spike with an absolute temporal distance larger than 3ms from the desired spike or fired more than one spike or did not fire at all, the classification was deemed incorrect. For example, if the neuron fired two spikes, one at 32ms and the other at 34ms, in response to a pattern from class 1 (with desired spike at 33ms), the input pattern is not considered as a class 1 pattern. This is a restricted criterion designed to assess the precise performance of the neuron. However, because the task is a classification one, wherein the classification accuracy is more important than the precise time of spiking, a more flexible approach is followed here. This approach is based on computing the difference between the neuron's actual response and the desired response using Eq. 11, then assigning the pattern to the class that produces the smallest error. For the above mentioned example, the pattern causing the neuron to fire two spikes at 32ms and 34ms will be labeled as class 1 provided that other neurons produced a response with a larger error.

Given the above two points, a number of approaches, based on the time of the firing spikes and the criterion to decide the class label, are valid for spatiotemporal spike pattern association using multiple SPANs. These approaches are evaluated in the next section.

4. Spike Pattern Classification and Spike Pattern Generation with Multiple SPAN

4.1. Experimental Setup

We follow the initiative recently proposed in [26] for promoting reproducible descriptions of neural network models and experiments. This initiative suggests the use of specifically formatted tables outlining neural and synaptic models and their parametrization, *cf.* Table 1.

The multiple SPAN network shown in Fig. 3 is used in the simulation.

We employ 200 input neurons that stimulate the synapses of each output neuron. The spike trains for each input neuron are sampled from a uniform random distribution in the interval $[0, 200]$ms. For simplicity, we allow only a single spike for each input neuron. Each output neuron is fully connected to all of the 200 input neurons with randomly initialized connection weights.

The same training and testing patterns generated in [1] are utilized here so that comparison may be drawn. The training patterns comprise of 5 classes, each with 15 samples generated by adding a Gaussian jitter with a standard deviation of 3ms to a randomly created base pattern. The testing set consists of $25 \times 5 = 125$ spike patterns generated in the same way [1]. Only the training set is used during training, while the testing set is used to determine the generalization ability of the trained network. The spike time of the output neuron encodes the class label of the presented input pattern. We allow 200 epochs for training and we repeat the experiment in 30 independent runs. For each run, a different set of random initial weights is selected.

All of our experiments employ the SNN simulator NEST [29].

4.2. Methods for Encoding Class Labels as Output Spike Sequences and Experimental Results

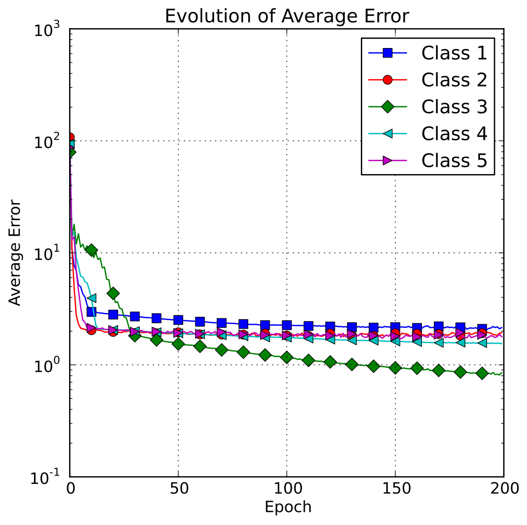
As pointed out in section 3 the class label from the spike output of the neurons can be determined by several means. We evaluate these approaches in the following tests:

Method 1: Multiple SPANs firing at different time instances. In [1] a single SPAN is trained to classify five classes of spatiotemporal spike patterns by firing at different time instances, namely 33, 66, 99, 132 and 165ms. The classification accuracy of that experiment is shown in the first row of Table 2.

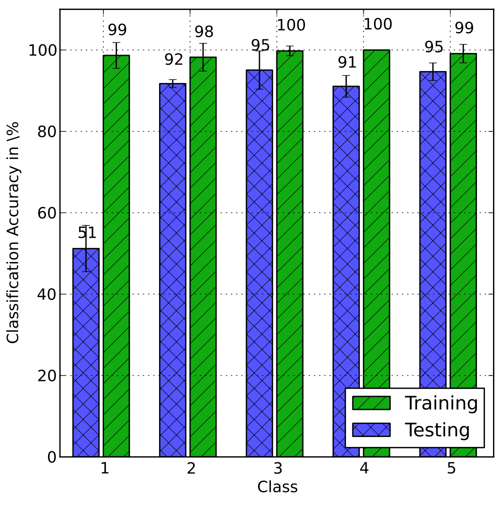
We repeat the experiment using the multiple SPANs architecture shown in Fig. 3. Each neuron is trained to fire a single spike at one of the specified times, {33, 66, 99, 132, 165} ms, to recognize a class. A pattern is deemed to belong to a specific class if the neuron fires a single spike within 3ms of its target spike. The results of this experiment are shown in Fig. 4b and are summarized in Table 2. Obvious improvement in the classification accuracy is obtained when multiple SPANs are used, especially in the testing phase. The testing accuracy is raised above 90% level for all of the classes except class 1 (which gained an improvement of 4% over

Table 2: Comparison of classification results using a synthetic benchmark problem. See the text for details on the different test scenarios investigated for the Multiple SPAN. Shown are the accuracies of the testing and training data (training accuracies are in brackets).

Method	Class					Average
	1	2	3	4	5	
Single SPAN [1]	47% (81%)	92% (100%)	87% (100%)	78% (100%)	94% (93%)	80% (94%)
Multiple SPAN Method 1	51% (99%)	92% (98%)	95% (100%)	91% (100%)	95% (99%)	84% (99%)
Method 2	99% (100%)	92% (100%)	81% (99%)	86% (100%)	94% (99%)	90% (100%)
Method 3	99% (100%)	96% (100%)	96% (99%)	93% (100%)	99% (100%)	96% (100%)



(a)



(b)

Figure 4: Spatio-temoral classification results using multiple SPANs. (**a**) Evolution of the average errors computed using Eq. 11. (**b**) The average accuracies obtained.

the single neuron accuracy). The over all accuracies for the training and testing phases are 99% and 84.8% respectively compared with 94.8% and 79.6% for the single neuron case. Fig. 4a quantifies the time difference between the output spikes and the desired spikes computed from Eq. 11. Although at the end of the training epochs the neurons do not spike precisely at the desired times, the training is performed correctly, evidenced by the classification accuracy approaching 100% during training. The results of the test show that using multiple SPANs improves the performance over a single SPAN.

Method 2: Multiple SPANs firing at single time instance. In this experiment all neurons are trained to fire a single spike at 165ms in response to a pattern belonging to the assigned class. If a neuron fires a single spike within 3ms of the target spike (165ms) the pattern is labeled with the class of that neuron. For this test, the obtained training accuracies are $\{100\%, 100\%, 99\%, 100\%, 99\%\}$ and the testing accuracies are $\{99\%, 92\%, 81\%, 86\%, 94\%\}$ for the five classes respectively. The overall testing accuracy of classification has improved from 84.8% to 90.4%. This is due to the increase in the classification accuracy of the first class from 51% to 99%, because of the timing of the target spike has been shifted from 33ms to 165ms. However, the time shift of the target spikes for classes 3 and 4 causes a slight drop in the accuracy of these classes. Overall, this test highlights the importance of proper timing of the desired spikes in improving classification accuracy.

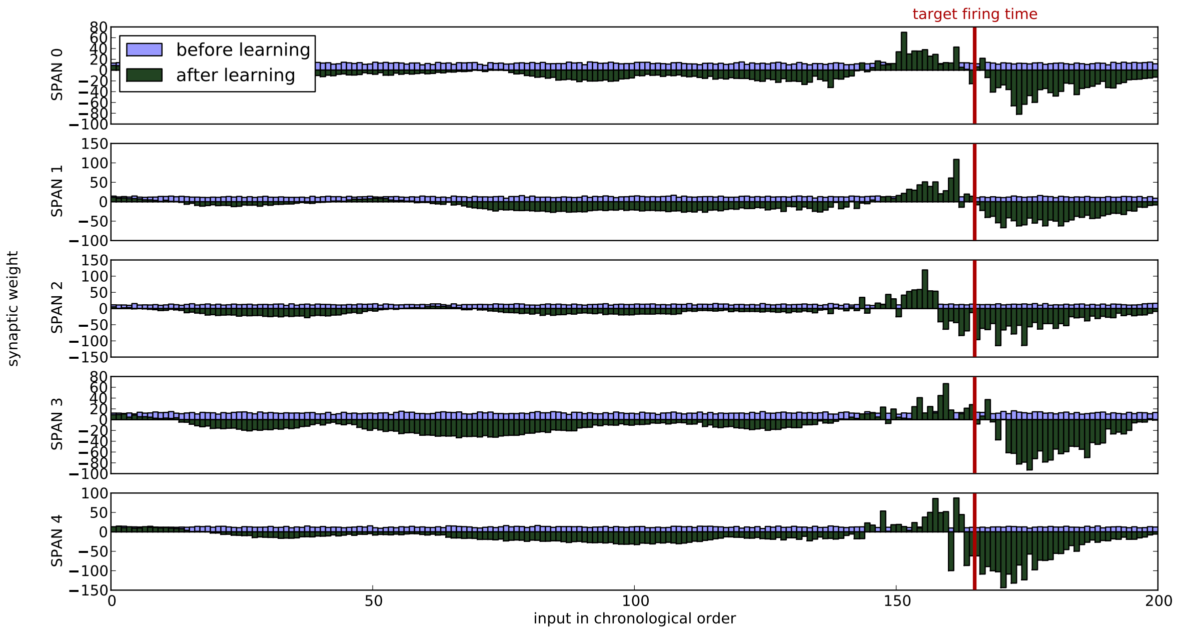


Figure 6: Evolution of the synaptic weights before and after training.

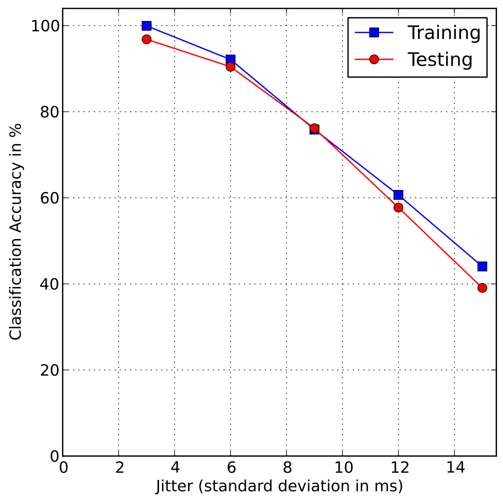


Figure 5: Evolution of the accuracy vs. adding more jitter to the training and testing patterns.

Method 3: Multiple SPANs firing at single instance with Eq.11 to determine the class label. Test 2 is repeated, but the label of the pattern is identified by applying Eq.11 to the output spike response of the neuron. The neuron that produces the minimum error is used to label the input pattern. The obtained accuracies of the training patterns are $\{100\%, 100\%, 99\%, 100\%, 100\%\}$ while those of the testing patterns are $\{99\%, 96\%, 96\%, 92.8\%, 99\%\}$. Thus, using Eq.11 to identify the class rather than the 3ms time difference has improved the testing accuracy from 90.4% to 96.6%.

The results of these three tests are summarized in Table 2.

Method 4: Multiple SPANs with more complex stimulus. This test evaluates the network using more complex patterns. The complexity of the input patterns is increased by increasing the amount of the time jitter added to the spikes (see section 4.1). The spikes of the patterns (both testing and training) are shifted by adding a jitter drawn from Gaussian distributions with different standard deviations. The criterian based on Eq. 11 is used to assign the class label from the neuron output. The overall training and testing classification accuracies as a function of the added jitter are plotted in Fig. 5. With a mean jitter of 6ms, the neurons retain testing accuracy above 90%. At jitter of 9ms the accuracy drops to 76%, and declines to below 40% at 15ms jitter.

To visualize the effect of the learning algorithm on the synaptic efficacy of SPANs, the average weight distribution before and after training for each of the five SPANs is illustrated in Fig. 6. Neural inputs are averaged and chronologically sorted according to their spike firing times. Each bar in the figure reflects the synaptic strength of a particular synapse of a SPAN trained on samples of a specific class. To gain an impression of the temporal causality of the weight changes, we overlay the plot with the desired firing times of the neuron (red vertical lines at 165ms in each plot). The figure presents the weight changes averaged over all 30 runs.

As discussed in the experimental section, the synapses are assigned positive weights distributed uniformly in the interval $[0, 25]$pA at epoch 0. After the training, in epoch 200, the weights assume Gaussian-like distributions especially around the target spikes, and some weights become negative. As expected, synapses that transfer input spikes which are temporally close to the desired target spikes are potentiated. On the other hand, synapses that transfer spike inputs at undesired times are inhibited. The high proportion of negative weights following tranining implies that the inhibition effect is rather strong. In the presented benchmark study, this observation did not adversely impact on the classification performance. However, future studies should investigate the configuration of the learning rate carefully to counteract an overly influence of the weight update rule devined in Eq. 10.

5. Conclusion and Future Work

Learning is an important process that allowes intelligence to emerge, empowering living entities to perform their natural daily activities. Although, the mechanisms of how learning reflects physiological change at the neuron level are complex, artificial neural networks mimic this process by iteratively adjusting synaptic weights in the direction of gradient of error function that quantifies the difference between the actual output behavior and the desired behaviour. Similarly, SPAN learning method for SNN [1] is based on minimising the difference (error) between a teacher spike signal and the actual spike signal to train a spiking neuron to recognize spatiotemporal spike patterns.

In this article we have applied SPAN [1] learning rule to a multiple-neuron network in which, each neuron is trained to recognize a single class rather than a single neuron being trained to recognize all classes. Thus, the burden of the learning task is distributed among the neurons. A single neuron is limited in its capacity to memorize and recognize spike patterns. Multiple neurons were shown to perform more accurately accuracy than a single one. Using spike timing to encode information,(such as class label) provides more flexibility and more options for temporal classification. Hypothetically, a single neuron can be trained to classify many classes, and is limited only by its memory capacity. In contrast, a neuron with binary output can recognise only two classes, characterised by the firing or non firing of a spike. Furthermore, temporal encoding suggests different ways to perform training.

In the multiple neurons experiment of section 4 each neuron was trained to patterns of a single class, *i.e.* each neuron was locked onto one class and when excited by a different pattern from a different class, the neuron is highly unlikely to fire accurately. When two samples belonging to two different classes are presented, the label of the sample will be decided by the closeness to the desired spikes. However, this response might depend on the stimulus and the application. In fact, more options are available to perform the classification, for example the neuron may be trained to produce a spike train rather than a single spike. Another option is increasing the number of neurons assigned to a class, and training each neuron to fire at a different time. These options increase redundancy and may help to accommodate the temporal structures of the stimulus.In all experiments in this paper we talked about classification of spatio-temporal input patterns and how the desired class label can be represented (as a single spike or trains of spikes at different times). But the scope of interpreting these methods and experiments is broader. The methods can be used to model complex systems of motor control as a result of perception recognition and classification of input stimuli.

Our future work is to apply the network to a real-world scenarios. In addition to the batch mode learning, incremental learning is being developed [30], *i.e.* weights are adjusted after each incoming spatio-temporal pattern rather than waiting until the end of presenting all the patterns.On-line learning is also being experimented, where weights would change at each input spike rather than after the whole pattern is presented. Hardware implementation of the SPAN algorithm on the INI SNN chip [31] is being tested. A challenging application problem is the use of the SPAN algorithm for motor control to achieve smooth control of neuro-prosthetics [32] and neuro-rehabilitaion robots [33]. In this case brain signals are measured and a SPAN-based system is trained to generate spike sequences to control a device for a complex, smooth movement. The prospect of using the recently proposed Neurogenetic Brain Cube (NeuCube) [34] that can learn to map brain signals into a 3D SNN Cube, and then use SPAN for recognising the NeuCube spatio-temporal spiking patterns and for generating spiking sequences for control is very promising. The idea is that instead of measuring brain spikes in an invasive way as in [32] and using them to control an object and movement, to create a brain model of the subject in a NeuCube, to train the NeuCube on different brain signals form the subject (e.g. EEG. MEG, etc) and then to train an output SPAN model to transfer the NeuCube internal spiking sequences into control signals.

Acknowledgment

This project is supported by the Knowledge Engineering and Discovery Research Institute (KEDRI, http://www.kedri.info) of the Auckland University of Technology. In addition, AM is supported by a grant from the NZ Ministry of Science and Innovation and NK by an EU FP7 Marie Curie project EvoSpike, hosted by the Institute for Neuroinformatics at ETH/UZH Zurich (http://ncs.ethz.ch/projects/evospike). The authors would like to thank Dr. Leonie Pipe for improving the paper.

References

[1] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, in: L. S. Iliadis, C. Jayne (Eds.), EANN/AIAI (1), volume 363 of *IFIP Publications*, Springer, 2011, pp. 219–228.

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

[3] W. Maass, Neural Networks 10 (1997) 1659 – 1671.

[4] W. Maass, C. M. Bishop (Eds.), Pulsed Neural Networks, MIT Press, Cambridge, MA, USA, 1999.

[5] W. Maass, in: Pulsed neural networks, MIT Press, Cambridge, MA, USA, 1999, pp. 55–85.

[6] J. Hopfield, Nature 376 (1995) 33–36.

[7] S. M. Bohte, NATURAL COMPUTING 3 (2004) 195–206.

[8] C. C. Bell, V. Z. Han, Y. Sugawara, K. Grant, Nature 387 (1997) 278–281.

[9] G.-q. Bi, M.-m. Poo, J. Neurosci. 18 (1998) 10464–10472.

[10] R. Legenstein, C. Naeger, W. Maass, Neural Computation 17 (2005) 2337–2382.

[11] T. Masquelier, R. Guyonneau, S. J. Thorpe, PLoS ONE 3 (2008) e1377.

[12] R. Gutig, H. Sompolinsky, Nature Neuroscience 9 (2006) 420–428.

[13] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Nature 323 (1986) 533–536.

[14] S. M. Bohte, J. N. Kok, J. A. L. Poutré, in: ESANN, pp. 419–424.

[15] F. Ponulak, ReSuMe – New supervised learning method for Spiking Neural Networks, Technical Report, Institute of Control and Information Engineering, Poznań University of Technology, Poznań, Poland, 2005.

[16] F. Ponulak, Applied Mathematics and Computer Science 18 (2008) 117–127.

[17] F. Ponulak, A. Kasiński, Neural Computation 22 (2010) 467–510. PMID: 19842989.

[18] W. Maass, T. Natschläger, H. Markram, Neural Computation 14 (2002) 2531–2560.

[19] F. Ponulak, A. J. Kasinski, in: Proceedings of EPFL LATSIS Symposium 2006, Dynamical Principles for Neuroscience and Intelligent Biomimetic Devices, Lausanne, Switzerland, pp. 119–120.

[20] R. V. Florian, The chronotron: a neuron that learns to fire temporally-precise spike patterns, Available from Nature Proceedings: http://precedings.nature.com/documents/5190/version/1, 2010.

[21] J. D. Victor, K. P. Purpura, Network: Computation in Neural Systems 8 (1997) 127–164.

[22] A. Mohemmed, S. Matsuda, S. Schliebs, K. Dhoble, N. Kasabov, in: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 2969 –2974.

[23] M. Tsodyks, K. Pawelzik, H. Markram, Neural Comput. 10 (1998) 821–835.

[24] B. Widrow, M. Lehr, Proceedings of the IEEE 78 (1990) 1415 –1442.

[25] M. C. van Rossum, Neural Computation 13 (2001) 751–763.

[26] E. Nordlie, M.-O. Gewaltig, H. E. Plesser, PLoS Comput Biol 5 (2009) e1000456.

[27] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, in: International Conference on Neural Information Processing, ICONIP 2011, Springer Verlag, Shanghai, China, 2011, pp. 718–726.

[28] W. Maass, H. Markram, Temporal integration in recurrent microcircuits, MIT Press, 2nd edition edition, pp. 1159–1163.

[29] M.-O. Gewaltig, M. Diesmann, Scholarpedia 2 (2007) 1430.

[30] A. Mohemmed, N. Kasabov, in: IEEE World Congress on Computational Intelligence , WCCI 2012, Brisbane, Australia, pp. 1227–1232.

[31] S. Moradi, G. Indiveri, in: Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE, pp. 277–280.

[32] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, J. P. Donoghue, Nature 485 (2012) 372–375.

[33] X. Wang, Z.-G. Hou, A. Zou, M. Tan, L. Cheng, Neurocomputing 71 (2008) 655 – 666.

[34] N. Kasabov, in: Artificial Neural Networks and Pattern Recognition, ANNPR 2012, Trento, Italy, p. To Appear.

Vitae

Ammar Mohemmed received BE, ME, and Ph.D in 1995, 2001, 2008 recepictively. Currently, he is working as a research fellow in the Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, New Zealand. His current research is Spiking Neural Network and its applications especially for image and computer vision.

Stefan Schliebs received a MSc in Computer Science at the University of Leipzig, Germany in 2006. and a PhD in 2010 at the Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland, New Zealand, under the supervision of Prof Nikola Kasabov and Dr Michaël Defoin-Platel. His research interests include evolutionary algorithms with a focus on Estimation of Distribution Algorithms and novel connectionist systems especially in the area of spiking neural networks. He is currently working as a postdoctoral research fellow at KEDRI on spatiotemporal pattern recognition problems using spiking neural networks.

Satoshi Matsuda received B.E., M.E., and Ph.D. degrees in 1971, 1973, and 1976, respectively, from Waseda University, Tokyo, Japan. Since 2000, he has been a professor of computer science at the Department of Mathematical Information Engineering, College of Industrial Technology, Nihon University, Japan. His main research interests include neural networks, neuroevolution, symbolic artificial intelligence, machine learning, and cognitive science. Professor Matsuda is a member of the IEEE(Computer Society and Computational Intelligence Society), the INNS, and the ACM. From 2002 to 2007 he served as an Associate Editor of IEEE Transactions on Neural Networks.

Nikola Kasabov (IEEE M'93 SM'98 F2010) obtained his Masters degree in computing and electrical engineering (1971) and PhD in mathematical sciences (1975) from the Technical University of Sofia, Bulgaria. He is the Director and the Founder of the Knowledge Engineering and Discovery Research Institute (KEDRI, ww.kedri.info) and Professor of Knowledge Engineering at the School of Computing and Mathematical Sciences at the Auckland University of Technology, New Zealand. Before that he worked as Professor at the University of Otago, Senior Lecturer at the University of Essex, the UK and Associate Professor at the Technical University Sofia. He has published more than 480 papers, books and patents in the areas of information science, computational intelligence, neural networks, bioinformatics, neuroinformatics. Prof. Kasabov is a Past President of the Intern. Neural Network Society (INNS) and the Asia Pacific Neural Network Assembly (APNNA). He is a Distinguished IEEE CIS Lecturer , an EU FP7 Marie Curie Fellow in the Institute for Neuroinformatics of the University of Zurich and ETH, and a Guest professor at the Shanghai Jiao Tong University. He has served as a chair and a program committee member of numerous IEEE, INNS, ICONIP, ANNES, NCEI and other international conferences. He is a Co-Editor-in-Chief of the Springer Evolving Systems journal and Associate Editor of several other journals, including Neural Networks and Information Sciences.