ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

# Spiking AGREL

Davide Zambrano[1], Jaldert Rombouts[2], Cecilia Laschi[1], and Sander Bohte[2]
[1] The BioRobotics Institute, Scuola Superiore Sant'Anna, Italy
[2] CWI, Amsterdam, The Netherlands

**Abstract**. Spiking neural networks are characterised by the spiking neuron models they use and how these spiking neurons process information communicated through spikes – the neural code. We demonstrate a plausible spiking neural network based on Spike Response Models and predictive spike-coding. When combined with a plausible reinforcement learning strategy – Attention Gated REinforcement Learning (AGREL), we show that such predictive spiking neural networks can compute non-linear mappings, including XOR. Our *spiking AGREL* achieves similar performance as standard AGREL, with much more efficient neural coding.

## 1 Introduction

Spiking neural networks are characterised by the spiking neuron models they use and how these spiking neurons process information communicated through spikes – the neural code. Ideally, the spiking neuron models and associated neural code are biologically plausible, and neural computation in such networks should offer both competitive performance for pattern recognition as well as insights into the workings of real biological neural networks.

In this paper, we present a plausible spiking neural network based on Spike Response Models and predictive spike-coding. When combined with a plausible reinforcement learning strategy – Attention Gated REinforcement Learning (AGREL) [1, 2], we show that for the first time, such spiking neural networks based on predictive spike-coding can compute non-linear mapping, including XOR. Related work, like [3, 4], only computes using a single layer of spiking neurons, and hence cannot compute non-linear mappings. Spiking neural networks based on spike-time coding, like [5, 6] can compute non-linear functions, but such schemes are hard to extend to time-continuous online computation.

In predictive spike-coding [7, 8, 9, 10], spiking neurons approximate an analog Artificial Neural Network (ANN) that operates in continuous time: the analog time-continuous signals that neurons in an ANN communicate, are approximated as a sum of spike-triggered kernels. The main advantage of such an approximate neural coding scheme is that it vastly reduces the bandwidth required to communicate ongoing neural activations. For large and deep neural networks, neural communication dominates the computational complexity [11], and efficient approximations are crucial for extending deep learning approaches to time-continuous operation, e.g. for video processing and dynamic control.

The next section describes the spiking neuron model framework, the network architecture and the reinforcement learning strategy used. Then, the simulation results on two non-linear mapping problems demonstrate the efficiency gain obtained with respect to the baseline results of a non-spiking network as a comparison among different threshold adaptation mechanisms.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

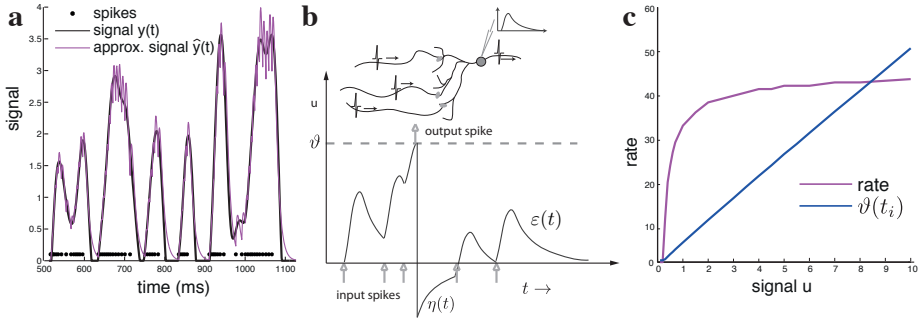


Figure 1: (a) Example of signal approximation with predictive spike-coding. (b) Spike Response Model. (c) Effective transfer function with multiplicative adaptation. Purple: firing rate as function of input current, blue: effective adapted threshold $\vartheta(t)$.

## 2 Feedforward Artificial Neural Networks

A standard neural network model computes the activation of each neuron $j$ as a function of the weighted sum of inputs originating from other neurons $i$:

$$x_j = \sum\nolimits_i w_{ij} y_i \qquad y_j = \mathcal{F}(x_j)\,,$$

A standard fully connected feedforward neural network is defined for input layer $\mathcal{I}$, hidden layer $\mathcal{J}$, and output layer $\mathcal{K}$, populated with neurons $i$, $j$ and $k$. A bias neuron is also added on each layer with its own weight $w_{0j}$. The a-cyclical neural network thus described does not include a notion of time. A continuous time computation can be achieved in a time-sliced manner: for each time-slice $\Delta t$, an input vector $x_i(t)$ is presented as input in the network, and the output vector $y_k(t)$ is computed through the network.

### 2.1 Predictive Spike-Coding

In predictive spike-coding, we approximate a (positive) continuous time signal $y(t)$ as a sum of kernels $\varepsilon(t)$, shifted by respective spike times $t_i$:

$$y(t) \approx \hat{y}(t) = \sum\nolimits_{t_i} \varepsilon(t - t_i), \tag{1}$$

where $\varepsilon(t)$ is usually modeled as an alpha-function [12]. An example of such predictive coding is shown in Fig. 1a (and example alpha-functions $\varepsilon(t)$ in Fig 1b). The greedy approximation $\hat{y}(t)$ can be computed by straightforward thresholding, for example by tracking the difference $z(t) = y(t) - \hat{y}(t)$. When this difference exceeds a threshold value $\vartheta$, a spike $t_i$ is generated, and a kernel $\varepsilon(t - t_i)$ is subtracted from the estimate $\hat{y}(t)$:

$$z(t) = y(t) - \hat{y}(t) \qquad t_i = t \quad \text{if} \quad z(t) > \vartheta$$

If we allow the threshold $\vartheta$ to be dynamic, $\vartheta(t)$, the amount of input needed to elicit a spike is effectively changed. With such behavior, *adaptive* predictive spike-coding computes a scaled sum of kernels:

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

$$\hat{y}(t) = \sum\nolimits_{t_i} \vartheta(t_i)\varepsilon(t - t_i) \tag{2}$$

A number of recent papers have noted that such a greedy approximation mechanism mimics the behaviour of real spiking neurons [7, 8, 9, 10].

*Spike Response Models (SRM).* The computation of $z(t)$ naturally maps to the Spike Response Model formulation of spiking neurons [12]:

$$u_j(t) = \sum\nolimits_i \sum\nolimits_{t_i} w_{ij}\varepsilon(t - t_i) - \sum\nolimits_{t_j} \eta(t - t_j) \qquad t_j = t \quad \text{if} \quad u_j(t) > \vartheta,$$

where $u_j(t)$ describes the membrane potential of a neuron $j$, which is computed as the (weighted) sum of post-synaptic responses $\varepsilon(t)$ each triggered by pre-synaptic spikes $t_i$. From this, refractory responses $\eta(t)$ are subtracted, each triggered by spiking events $t_j$. As illustrated in Fig. 1b, in an SRM, a spike is triggered when $u(t)$ crosses a threshold $\vartheta$ from below. Predictive coding is recovered when we set $\varepsilon(t)$ to (low-pass filtered) $\eta(t)$.

While the SRM as described above is a relatively crude approximation to real spiking neurons, a substantially improved fit can be obtained when the threshold $\vartheta$ is also treated as a dynamic quantity, e.g.:

$$\vartheta(t) = \vartheta_0 + \sum\nolimits_{t_j} \gamma(t - t_j),$$

where $\gamma(t)$ is a kernel with exponential or power-law decay [13]; the refractory response $\eta(t - t_j)$ is then also scaled by the threshold, $\vartheta(t_j)$ [9]. In [9], it was shown that the correspondence with real spiking neurons can be increased by including a *multiplicative* dynamic threshold $\vartheta(t)$:

$$\vartheta(t) = \vartheta_0 + \sum\nolimits_{t_j} \vartheta(t_j)\gamma(t - t_j).$$

A multiplicative spike-triggered threshold adaptation function models fast spike-triggered adaptation processes that maximize information transmission [9]. Importantly for neural networks, when interpreted as predictive spike-coding, [9] showed that such predictive spike-coding efficiently encode neural signals over a vast dynamic range. Threshold adaptation thus addresses the problem that simple fixed-threshold predictive spike-coding implicitly requires that the signal range for any particular neuron is proportional to the threshold $\vartheta$. Note also that in such Spike Response Models, the transfer-function is effectively rectified linear, as, rather than with a firing rate, the sum of spike-triggered kernels $\hat{y}(t)$ is taken as the neural code [9] (Fig 1c).

### 2.2 Non-linear Plausible Reinforcement Learning: AGREL

Supervised learning algorithms like error-backpropagation can tune the weights in a standard multi-layer ANN to learn non-linear input-output mappings, like the famous XOR problem. In nature, the "right" answer is usually not available, and learning is often a form of Reinforcement Learning [14]. Roelfsema & Van Ooyen [1] introduced AGREL as a biologically plausible algorithm for reinforcement learning of direct reward. The AGREL network is shown in Fig. 2a: given

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

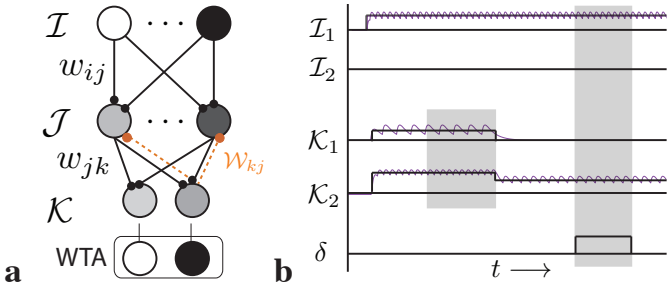


Figure 2: (a) AGREL network. (b) Actual neural processing for an example input-output. Input current is continuously injected into the input neurons. After some time (1st grey area), the controller computes the winning action-value neuron, and sets the activity of the winning output neuron to 1, and that of the losing output neuron(s) to 0. Some time later (2nd grey area) reward is received and the reward error $\delta$ is sent into the network for a brief period of time. Purple: predictive spike-coding approximation.

an input into a neural network, a value for each available action is computed by the action-value neurons in layer $\mathcal{K}$. A controller (e.g. in basal ganglia) then stochastically selects one of these actions, biased by the expected value. The winning action-value neuron's activity is then set to 1, and this winning activity is projected back to the hidden neurons weighted by weights $\mathcal{W}_{kj}$ (identical to forward weights $w_{jk}$). Given the selected action, a reward is then received and a reward error is computed: the difference between the reward and the expected value. Based on this reward error, the weights in the network, including those in the hidden layer, can be updated in a biologically plausible local manner to effectively carry out stochastic gradient descent. A time-continuous AGREL computation is shown in Fig. 2b. Importantly, AGREL can learn non-linear mappings like XOR, which require a hidden layer and neurons with a non-linear transfer function.

Formally [2], the winning action-value neuron $K$ has activation set to $y_K = 1$; each timestep, the output weights $w_{jK}$ and hidden weights $w_{ij}$ are updated as:

$$\Delta w_{jK}(t) \propto \delta(t) y_K(t) y_j(t) \qquad \Delta w_{ij}(t) \propto \delta(t) y_i(t) I^+[y_j(t)] w_{jK} y_K(t),$$

where $I^+(y_j)$ is the derivative of a rectified linear unit (1 for $y_j > 0$, 0 else).

*Spiking AGREL.* Given predictive spike-coding, *spiking* AGREL is an adaptation of AGREL. The input signal is injected as current into the SRM. The output from the respective pre-synaptic neurons is then computed through predictive spike-coding: $\hat{y}(t)$. This gives the following learning rules:

$$\Delta w_{jK}(t) \propto \delta(t) y_K(t) \hat{y}_j(t) \qquad \Delta w_{ij}(t) \propto \delta(t) \hat{y}_i(t) I^+[y_j(t)] w_{jK} \hat{y}_K(t).$$

The predictive-coding signal approximation to continuous-time AGREL problem is illustrated in Fig. 2b as the purple line.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

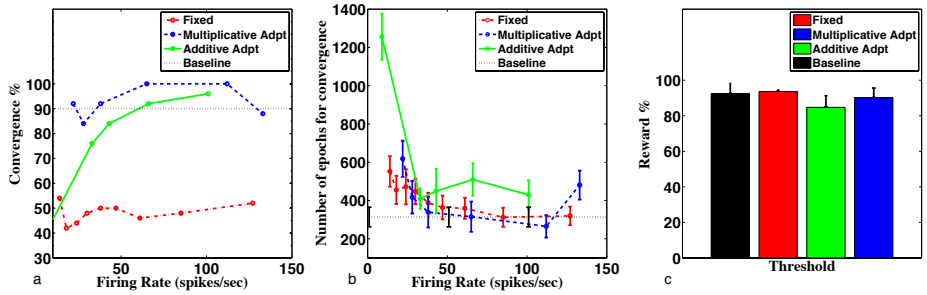


Figure 3: (a) XOR convergence rate. (b) Epochs required for convergence for the XOR problem. (c) Rate of reward for learning IRIS dataset. In both problems, the average firing rate in the spiking neural networks was varied either by setting the fixed threshold value, or the $Fp$ parameter for adapting spiking neurons.

## 3 Experiments

We evaluate the approximate spiking neural network on two non-linear problems: XOR and IRIS. For both problems, we computed the non-spiking AGREL solution to these problems in a continuous time mode as a baseline. We then compared this to spiking AGREL with the three types of dynamic threshold described before: fixed, additive adaptive and multiplicative adaptive.

*XOR.* Two binary inputs are presented to the network. The first output neuron fires when both inputs are equal ($1,1$ or $0,0$). The second fires in the other cases ($0,1$ or $1,0$). Ten hidden layer neurons are used in this task. For the adaptive thresholds, the resting threshold $\vartheta_0$ was set to 0.025, and $\gamma(t)$ decayed with power-law constant of $\beta = -0.85$ (after [13]), as $F_p(t-t_i+0.2)^{-\beta}$, and response kernels $\eta(t)$ and $\varepsilon(t)$ were modeled as $\exp(-t/20)$ and $1.45(1-\exp(t/4))\exp(-t/20)$ respectively (time $t$ in ms). A bias neuron has been added for both the input and the hidden layer. Each threshold method was tested on 25 runs with randomly initialized weights. For each run, the four input pattern were shown to the network for 2000 epochs. The convergence criterium was met when 80% of the answers were correct in a time window of 100 epochs.

*IRIS.* The Iris data set is a classical non-linear mapping problem. The data set contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. For this problem, the resting threshold $\vartheta_0$ was set to 0.2, with $F_p$ set to 45 for additive and 2.5 for multiplicative threshold to obtain the same maximum firing rate of 90 spikes/second for both spiking neuron models. Fig. 3c shows the obtained reward after 1000 epochs divided by the maximum possible reward, averaged over 20 runs.

Spiking AGREL was able to learn the non-linear mapping required for both XOR and IRIS problems, with the multiplicative adaptation method achieving performance similar to the baseline (Fig. 3)a,c; for XOR, higher firing rates generally perform better than lower ones (Fig. 3a), while not having a significant impact on convergence time (Fig. 3b).

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

## 4 Conclusion

We demonstrated how a spiking neural network with predictive spike-coding can approximate a time-continuous ANN efficiently like in [9]. Importantly, we showed that the resultant "spiking" AGREL can learn non-linear mappings as well as AGREL, particularly with multiplicative adaptation. The latter effectively fixes the spike-rate that a spiking neuron uses to communicate significant signals, almost irrespective of the size of the signal. Thus, and in contrast to spiking neurons with a fixed threshold, small signal values can be communicated as well as large signals. We additionally demonstrated how adaptive predictive spike-coding can be combined with a biologically plausible learning rule. In terms of bandwidth used, spiking neurons potentially are much more efficient compared to traditional ANN: communication is simplified by transmitting only one bit of information every time-slice rather than an analog value. Additionally, weight multiplications are only needed when a spike is received, saving tremendously on computational complexity. Together, this makes predictive spike-coding in spiking neural networks also highly suitable for scaling to large-scale simulations.

## References

[1] P.R. Roelfsema and A. Ooyen. Attention-gated reinforcement learning of internal representations for classification. *Neural Computation*, 17(10):2176–2214, 2005.

[2] J.O. Rombouts, A. van Ooyen, P.R. Roelfsema, and S.M. Bohte. Biologically plausible multi-dimensional reinforcement learning in neural networks. In *ICANN 2012*, pages 443–450, Berlin, Heidelberg, 2012. Springer-Verlag.

[3] R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nature*, 9(3):420–428, 2006.

[4] R. Urbanczik and W Senn. Reinforcement learning in populations of spiking neurons. *Nature neuroscience*, 12(3):250–252, 2009.

[5] S.J. Thorpe and J. Gautrais. Rank order coding. *Computational Neuroscience: Trends in Research*, 13:113–119, 1998.

[6] S.M. Bohte, J.N. Kok, and H. La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48:17–38, 2002.

[7] S.M. Bohte and J.O. Rombouts. Fractionally Predictive Spiking Neurons. In *NIPS 23*, pages 253–261, 2010.

[8] M. Boerlin and S. Denève. Spike-based population coding and working memory. *PLoS computational biology*, 7(2):e1001080, February 2011.

[9] S.M. Bohte. Efficient spike-coding with multiplicative adaptation in a spike response model. In *NIPS 25*, pages 1844–1852, 2012.

[10] D. Chklovskii and D. Soudry. Neuronal spike generation mechanism as an oversampling, noise-shaping a-to-d converter. In *NIPS 25*, pages 512–520. 2012.

[11] L. Slazynski and S.M. Bohte. Streaming parallel gpu acceleration of large-scale filter-based spiking neural networks. *Network: Computation in Neural Systems*, 23:183–211, 2012.

[12] W. Gerstner and W. Kistler. *Spiking neuron models*. 2002.

[13] C. Pozzorini, R. Naud, S. Mensi, and W. Gerstner. Temporal whitening by power-law adaptation in neocortical neurons. *Nature neuroscience*, 2013.

[14] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. 1998.