U.P.B. Sci. Bull., Series C, Vol. 69, No. 3, 2007 ISSN 1454-234x

TRAINING SPIKING NEURONS WITH ISOLATED SPIKES CODING

B. PĂVĂLOIU[1], P. CRISTEA[2]

Lucrarea prezintă structura şi funcţionarea reţelelelor neurale cu impulsuri. Sunt descrise principalele modele, precum şi modalităţile în care se reprezintă şi procesează datele. Pentru un neuron de tipul „integrează şi declanşează" se foloseşte o codare a datelor de intrare bazată pe temporizare. Este descrisă o metodă de antrenare supervizată pentru acest tip de neuron şi se demonstrează că poate fi găsit un echivalent de tipul perceptron/ învăţare perceptron.

The paper presents the structure and function of spiking neural networks. There are described the main models, as well as the modalities of data representation and processing. For an "Integrate-and-Fire" neuron, it is used a timing coding of input data. It is described a method of supervised training for this type of neuron and it is proved that an equivalent perceptron/ perceptron training rule can be found.

Keywords: spiking neural network, integrate-and-fire, spiking response model, coding, timing coding, supervised training, perceptron training rule

1. Introduction

In the general framework of Artificial Intelligence, the connectionist systems play an important role. The term of Neural Network is used instead of Artificial Neural Network (ANN), despite the fact that the biological start has been mainly a source of inspiration than a model for the connectionist soft computation. While a "natural" Neural Network is a group of biological neurons acting together, an ANN is system that tries to perform intelligent calculus tasks, emulating the structure and functions of real neurons. Unfortunately, even if the quantity and the quality of the data about the structure and low-level functioning of biological neural systems are increasing continuously, the knowledge about the way information is represented and processed is still small and unconvincing.

A biological neuron receives action potential (spikes) from the afferent neurons and, after their processing as inputs, it can issue a spike to participate at the activation of the efferent neurons.

[1] Assist Prof., Faculty of Engineering in Foreign Languages, University POLITEHNICA of Bucharest, Romania
[2] Prof., Faculty of Electrical Engineering; University POLITEHNICA of Bucharest, Romania

Even if there are several thousands of neuron types, their structure is essentially similar, designed to sum in soma the input information from other neurons, whose axons are connected to its dendrites through synapses, and when certain conditions are fulfilled, to trigger an action potential that will travel down the axon to the synapses with other neurons.

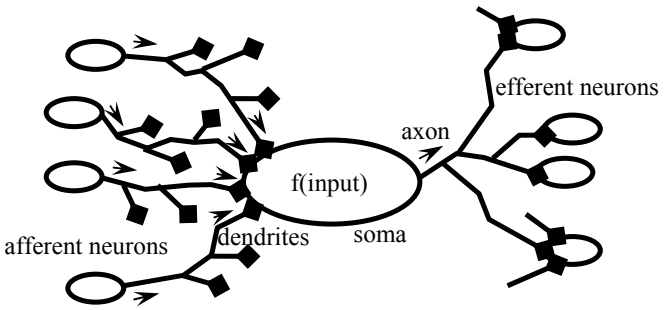

Fig. 1. Neuron structure and links

Wolfgang Maass [7] classifies ANNs in three generations.

The first generation consists of networks that have as basic computation element the McCulloch-Pitts neuron. Their outputs are binary, being "high" if the sum of the weighted inputs exceeds a threshold and "low" otherwise. It has been shown that the feed-forward networks from the first generation with at least two layers (one "hidden" layer) can represent any boolean function, after it is written in conjunctive/ disjunctive form [8]. Some prestidigious representatives of the first generation are also the Hopfield networks and the Boltzmann machines.

The second generation is made of the networks that have as output a continuous function. The most used are the feed forward, the recurrent and the radial basis function networks. It was shown [3] that the feed-forward networks from the second generation, with sigmoid activation function and at least two layers can represent (approximate with any given precision) a continuous function on a compact interval.

ANN from the second generation can store, in its nonlinear mapping complex relations relating the output to the input. The back-propagation algorithm is the most frequently used algorithm for ANN supervised learning and implicitly for the second generation of NN. It has the advantages of parallel processing and simple implementation and its association with the feed forward NNs constitutes probably the most used paradigm in intelligent calculus.

The resemblance in the functioning of the first two generations of NN to the biological ones resides in the activation of a unit when certain combinations of its inputs exceed a given threshold. Neuron models of the first two generations do

not employ individual pulses, but their output signals typically lie between 0 and 1. These signals can be seen as normalized firing rates (frequencies) of the neuron within a certain period of time. This is a so-called rate coding, where a higher rate of firing correlates with a higher output signal. Rate coding implies an averaging mechanism, as real spikes work binary: spike, or no spike, there is no intermediate. Because of the averaging window mechanism the output value of a neuron can be calculated iteratively. After each cycle involving all neurons, the 'answer' of the network to the input values is known. Real neurons have a base firing-rate (an intermediate frequency of pulsing) and continuous activation functions can model these intermediate output frequencies. Hence, neurons of the second generation are more biologically realistic and powerful than neurons of the first generation.

The third generation consists of the spiking neural networks, which are closer to the biological model and more plausible from neuro-physiological point of view. The computing elements of spiking NNs pass on temporal sequences of pulses (spikes). If the incoming spikes to a neuron have a good timing, their effects merge and a spike is emitted to the subsequent neurons. The spiking neuron will remain in a refractory state for a certain amount of time. In Table I we present some comparison terms with NN from the first generations.

Table 1

NN Generations comparison

Classical NN	**Spiking NN**
Blunt biological plausibility	Good biological plausibility
Rate coding	Different time dependent codings
Relative simple models	Complicated models
Concurrent and rarely recursive	Time dependent, can be highly recursive
Small to medium size	Small to large (10^6 elements) size
Unaltered data	Randomness and noise in the system
Robust, but not for time dependent problems	More robust for all problems
Full connectivity between layers	Small connectivity, full recurrence
Mainly supervised training	Until now, mainly unsupervised training
Simple to simulate on computers	Time-driven or event-driven difficult simulation
Digital implementation on computers	Simple analog VLSI implementation

The SNNs posses a level of recurrence higher than classical NN and they are more suited for discrete problems due to their nature [2].

2. Spiking neural networks
2.1 Threshold-Fire Models

The threshold-fire models are based on the temporal summation of all presynaptic neurons activities (called presynaptic potentials - PSP) to the membrane potential $u(t)$. If this potential exceeds a certain threshold ϑ, then the neuron will fire, participating at the activation of the subsequent neurons. The most known threshold-and-fire models are the ***integrate-and-fire*** and the ***spike response model*** – SRM [5].

The integrate-and-fire neuron is perhaps the most used and well-known example of a formal spiking neuron model. The basic model is also called leaky-integrate-and-fire (LIF) because the membrane is assumed to be leaky due to ion channels, such that after a PSP the membrane potential approaches again a reset potential u0t. This model is also known as the linear integrate-and-fire neuron.

The equivalent circuit modeling the cellular body activity is presented in Fig. 2 – an RC circuit is charged by the input current and if the voltage across the capacitor C attains a threshold ϑ, the circuit is shunted and a pulse is transmitted through the axon to the efferent neurons.

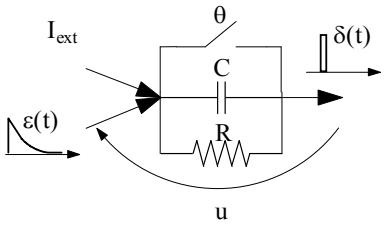

Fig. 2. Equivalent circuit for “Integrate and Fire” neuron model

The conservation of charge on soma leads to

$$C \cdot \frac{du}{dt} + \frac{u(t)}{R} = I(t) \tag{1}$$

The solution of this differential equation is:

$$u(t) = u_{rest} e^{-\frac{t-t_f}{\tau}} + \int_{t_f}^{t} \frac{I(s) e^{\frac{s-t}{\tau}}}{\tau} ds \tag{2}$$

The equation is easily dealt in finite difference form and is used for a time-driven simulation, computing the values of each neuron potential for each time step dt:

$$u(t+dt) = (1-\frac{dt}{RC})\cdot u(t) + \frac{dt}{C} I(t+dt) \tag{3}$$

Spiking Response Model (SRM)

In Fig. 3 is presented the equivalent circuit for inter-neural signal conduction (low-pass filtering).

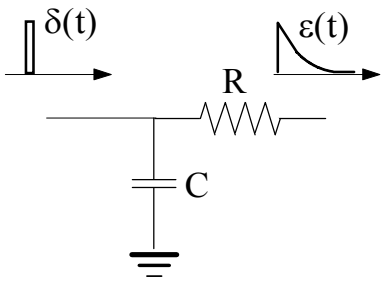

Fig. 3. Equivalent circuit for synapse

The spiking response model is defined in [6] and it describes the state of a neuron by a single variable *u*, using kernel functions for different aspects of its behavior. Its state is described as a summation of presynaptic pulse-response functions, a self-spike response function and an external input function.

For $\hat{t}_i$ the last firing time of the neuron i , the evolution of the neuron is given by:

$$u_i(t) = \eta_i(t-\hat{t}_i) + \sum_{j\in\Gamma_i} w_{ij} \sum_{t_j^{(f)}\in F_j} \varepsilon_{ij}(t-\hat{t}_i, t-t_j^{(f)}) + \int_0^\infty \tilde{\varepsilon}(t-\hat{t}_i, s) I^{ext}(t-s)ds \tag{4}$$

The SRM can be simplified presuming that the external (analog) current is 0 and neglecting the third term (any problem specific input data will be introduced by additional encoding neurons) and by neglecting the dependence of the ε_{ij} kernel on the term t-t_i (the effect of the neuron's last spike on the postsynaptic potential function)

$$u_i(t) = \eta_i(t-t_i) + \sum_{j\in\Gamma_i} w_{ij} \sum_{t_j^{(f)}\in F_j} \varepsilon_0(t-t_j^{(f)}) \tag{5}$$

This model is known as SRM0 and it is used largely for simulation purposes.

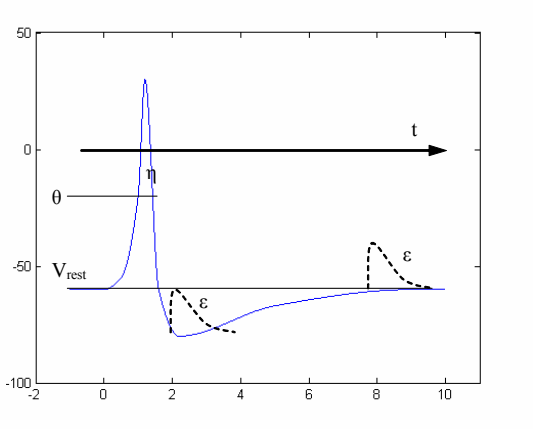

Fig. 4. Spike Response Model

The neuron fires if u_i reaches a threshold ϑ from below. The firing time $t_i^{(f)}$ is defined as the moment when $u_i(t) = \vartheta$ and $\frac{d}{dt}u_i(t) > 0$.

This model approach can be used to represent a large number of the other models. Due to its dependence only of time, it is used to avoid integration methods and time-driven simulation. Solving the kernel function equations that determine the nearest threshold-crossing point, corresponding with the next firing time, for each neuron, an event driven approach for the simulation can be applied.

Different functions are considered for ε, from simple and easy to be solved ones to others that are more plausible.

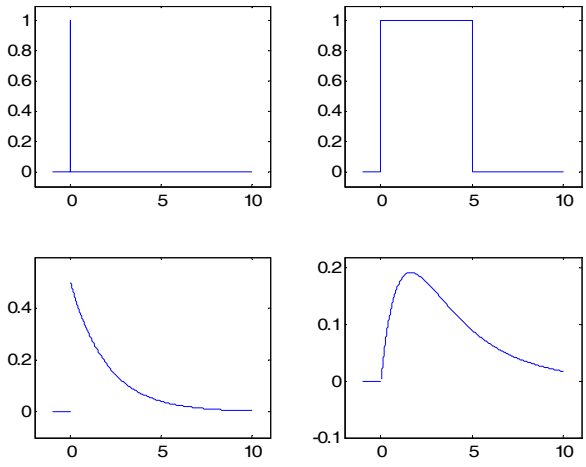

Fig. 5. PSP waveforms A- $\delta(t)$, B – square, C- exponential $e^{-\frac{t}{\tau}}$, D – alpha $te^{-\frac{t}{\tau}}$

3. Data representation in SNN

The way the data is encoded to enter the network plays a crucial role to the system. This is the spot few things are known in neuroscience and any advance here will push the things a lot further.

Rate code

This is the coding used by most of the neural system models and it assumes that the information is encoded as the firing rate of the neurons. There is a long history of this belief and there is evidence for it, starting with the experiments of Adrian nearly a century ago, that shown an increased firing rate for an increased intensity of the stimulus [1]. This experiments lead to the supposition that an entire train of spikes can be represented by a single number, representing the frequency of the spikes. Classical NN works with this numbers as the natural way of data representation. If an input has a higher value, it will have a higher "frequency" assigned and it will influence stronger the neuron.

Several methods data coding and representation in spiking NN are proposed in literature, having features as plausibility and representability. A good analysis is made in [10]. Each input neuron will fire at most one time.

Count code

Is a weaker version of rate coding. For a given time window and a number of N inputs, the coded value is the number of neurons that fire in the defined window. The number of states that can be defined is N+1.

Binary code

We separate the importance of the neurons from count code. A natural choice is to consider the value encoded by each neuron with a double importance than the precedent one. This way, the population of neurons works like a binary encoder, with the number of states 2^N.

Timing code

This coding presumes that we can determine the precise time of spiking for a neuron. If the number of "selecting" windows in the run interval is K, the number of states encoded by a neuron is K. Using N input neurons the number of states is K*N.

Synchrony based code

If consider the order of the input neurons, we take different significances for the input neurons, like in the binary coding, we will have K^N states. The spikes emitted by input neurons in the same time window are considered to be synchronic.

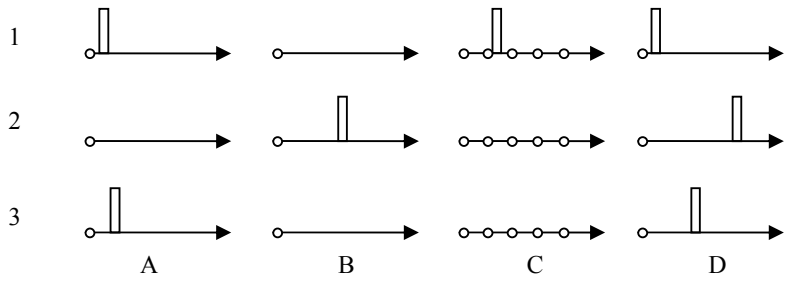

Fig. 6. Data representation of the number 2 using 3 input neurons. A – count code, B – binary code, representation of binary 10, C – timing code, second interval, D- rank order code - the second permutation 1-3-2

Rank order code

The code is given by the order the input neurons spikes. For N inputs, there are perm(N) = N! sequences in which the neurons can fire.

Thorpe and Gautrais [9] propose a nice circuit sensitive to the rank order activation of its inputs A-E. The neuron receives excitatory inputs from each of the inputs and shunting inhibition from an inhibitory neuron whose activity increases every time one of the inputs fires. As a result, only the first input to fire is unaffected by the shunting inhibition, and the inhibition increases progressively during the processing of a wave of spikes, each spike received by the neuron I making all the weights to be decreased by 50%. Thus for initial weights of 5, 4, 3, 2, 1 the total activation for the order of the spikes A, B, C, D, E is $5+4*0.5+3*0.5^2+ 2*0.5^3+1*0.5^4=8.06$.

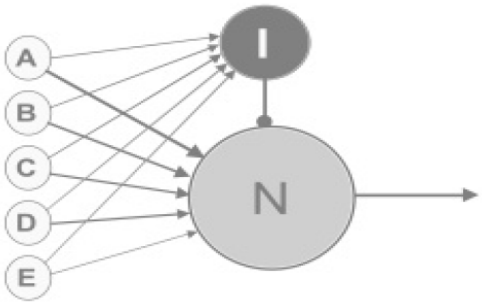

Fig. 7. SNN sensitive to rank order coding, from [9]

The final activation of the output neuron will be maximal only when the inputs are activated in the order of their weights and if the threshold is set to 8, the neuron will fire only when the sequence ABCDE is presented.

To reject the sequences where one input will act more than once, we can make the spike emitted from one input neuron to inhibit that neuron.

Delay coding

Because time is continuous and we have it included in the spike equation, we can encode with a single spike any analog number x as a spiking time x-T, where T is a given reference time. This version of timing coding, named in many papers as "delay coding", is used for problems addressed formerly by the NNs from the second generation.

4. Spiking neurons with isolated spikes coding

Suppose we have N symbols S_i, i=1:N. We want that each symbol to have its proper code, considered as a specific delay from 0. We will take for the sake of simplicity isolated spikes, i.e. spikes that have the PSPs non superposing for encoding different symbols. Since we are interested only in the first emitted spike, we will assume that the refractory period is very large, avoiding other spikes. Because the PSPs are identical for the same symbol, their addition will be in fact a weighted sum (corresponding to the number of occurrences) of the "basic" PSP, delayed to encode different symbols. For this reason, the waveform of the PSP doesn't matter, so it can be considered that ε_0 is the δ function - see (4) and Fig. 5.

A neuron will have R inputs p_1, p_2, p_3, …, p_R and one output t.

We have a set of Q input vectors and a set of Q targets with symbols from S, presented concurrently to the network. The input vectors P_1, P_2, …, P_Q, will have length R, and a set of M targets T_1, T_2, …, T_k, …, T_Q, $T_k \in S$

We will have to introduce for reference other N inputs to the network, each one emitting a spike with a certain delay, corresponding to the symbol it represents. The spiking neuron will have thus N+R inputs.

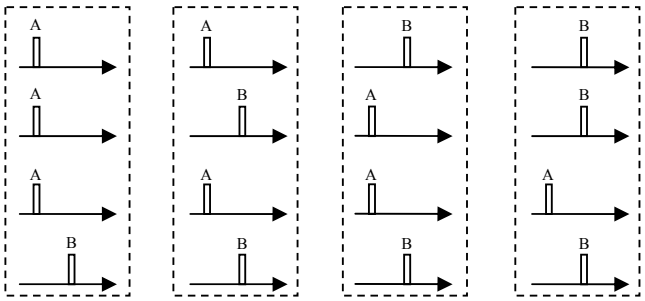

Fig. 8. Spike/PSPs coding the input data for symbols S1=A and S2=B; each box represents an input example

For a given pattern j, the equation (4) leads to:

$$u_j(t) = \sum_{i=1}^{N+R} w_i \varepsilon_{ij}(t) \tag{6}$$

For $p_{ij} = S_k$, we have $\varepsilon_{ij}(t) = \delta(t_k)$ and from there

$$u_j(t) = \sum_{i=1,}^{N} w_i \delta(t_k) = \sum_{k=1}^{M} \delta(t_k) \sum_{i|\, p_{ij}=S_k} w_i \tag{7}$$

If $T_j = S_k$, we have $U_j(t) = \delta(t_k)$, such that we need:

$$\sum_{i|\, p_{ij}=S_k} w_i \geq T \tag{8}$$

The threshold should not be reached before t_k, so:

$$\sum_{i|\, p_{ij}=S_l, l<k} w_i < T \tag{9}$$

The weights will sum up independently, so we devise the following algorithm:

For a given example, if the output is not the target symbol, increase all the weights for the inputs where the symbol corresponding to the target is present; if the output is a symbol with a prior time coding than the target symbol, decrease all the weights for the inputs where that symbol is present.

For 2 symbols, regarded here as the binary logical values S_1=0 and S_2=1, we show the weight and threshold values obtained with the described training algorithm for all the boolean functions of 2 variables.

Table 2

Weights and threshold for 2 inputs logical binary functions

Function	w_1	w_2	w_3	w_4	T
0	1	1	1	0	1
AND(x_1, x_2)	1	1	0	0	1
AND($x_1, \bar{x}_2$)	1	-1	1	0	1
x_1	1	0	0	0	1
AND($\bar{x}_1, x_2$)	-1	1	1	0	1
x_2	0	1	0	0	1
XOR(x_1, x_2)	-				
OR(x_1, x_2)	1	1	-1	1	1
AND($\bar{x}_1, \bar{x}_2$)	-1	-2	3	1	1
XOR($\bar{x}_1, \bar{x}_2$)	-				

Function	w_1	w_2	w_3	w_4	T
$\bar{x}_2$	1	-2	1	1	1
OR($x_1, \bar{x}_2$)	1	-1	0	1	1
$\bar{x}_1$	-2	1	1	1	1
OR($\bar{x}_1, x_2$)	-1	1	0	1	1
OR($\bar{x}_1, \bar{x}_2$)	-1	-1	1	2	1
1	0	0	0	1	1

The algorithm works, but we have to substantiate its behavior, including its failure to solve nonlinear separable problems.

In matrix form, we will represent a symbol S_k as the discrete set of amplitudes in moments t_k, k=1,N, with an element being one and the other zeros.

$$S_k = [0 \quad \cdots \quad \overset{k}{\hat{1}} \quad 0 \quad \cdots] \tag{10}$$

Let us study again the example j.

$$P_j = \begin{bmatrix} p_{1j} \\ \vdots \\ p_{Rj} \\ p_{(R+1)j} \\ \vdots \\ p_{(R+N)j} \end{bmatrix} = \begin{bmatrix} S_{f_j(1)} \\ \vdots \\ S_{f_j(R)} \\ S_1 \\ \vdots \\ S_N \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \cdots & \overset{f_j(1)}{\hat{1}} & 0 \\ & & & \\ 0 & \overset{f_j(R)}{\hat{1}} & 0 & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{N \text{ columns}} \tag{11}$$

The aggregate input Aj is:

$$A_j = [w_1 \quad \cdots \quad w_{R+N}] \underbrace{\begin{bmatrix} 0 & \cdots & \overset{f_j(1)}{\hat{1}} & 0 \\ & & & \\ 0 & \overset{f_j(R)}{\hat{1}} & 0 & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{N \text{ columns}} \tag{12}$$

The output of the neuron will be represented by the index of the first element in the vector A*j* larger than the threshold *T*.

$O_j = \min(i \mid A_j(i) >= T)$

If $T_j=S_k$, $\min(i \mid A_j(i) >= T) = k$ and from here

$A_j(i) < T$ for $i<k$, $A_j(k) >= T$ and it doesn't matter what happens for $i>k$.

For a perceptron with the same weights w_i, i=1:$R+N$ and the threshold T, this example is equivalent with the set of k examples with inputs $R_i=P_j(:,i)$, i=1:k, and the targets T_i=0, i=1:k-*1*, T_k=1.

We proved thus that the computing power of a spiking neuron with isolated spike codes is comparable with the one of a perceptron and determined the equivalent training patterns.

5. Conclusions

We presented the main features of spiking neural networks and of information encoding in SNNs. For a set S with N symbols $S=\{S_1, S_2, \ldots, S_N\}$ we presented an algorithm that will train a SNN with isolated spikes coding to represent a mapping $f:S^N \rightarrow S$, whether it is possible. We show that finding the appropriate weights is equivalent with a perceptron training problem and expose the corresponding training patterns.

REFERENCES

[1] *E. D. Adrian,* The basis of sensation, Norton, New York, 1928

[2] *Paul Dan Cristea*, *Bujor Păvăloiu*, Discrete event dynamic systems modeling using artificial neural networks, Rev. Roum. Sci. Techn. - Électrotechn et Énerg., **45**, p. 75-90, 2000

[3] *G. Cybenko*, Approximations by Superpositions of a Sigmoidal Function, Math. Cont. Signal \& Systems, **2** (1989) 303-314

[4] *Wulfram Gerstner*, Time structure of the activity in neural network models. Phys. Rev. E, **51**(1), pp. 738-758, 1995

[5] *Wulfram Gerstner,* What's different with spiking neurons? in Henk Mastebroek and Hans Vos, editors, Plausible Neural Networks for Biological Modelling, pages 23–48. Kluwer Academic Publishers, 2001.

[6] *Wulfram Gerstner and Werner M. Kistler*, Spiking Neuron Models, Single Neurons, Populations, Plasticity, Cambridge University Press, 2002

[7] *Wolfgang Maass*, Networks of spiking neurons: the third generation of neural network models, Neural Networks, **10**, pp. 1659-1671, 1997.

[8] *W. S. McCulloch, W. Pitts*, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, **5**, pp. 115-133, 1943

[9] *S. Thorpe and J. Gautrais*, Rank order coding, in Computational neuroscience: Trends in research 1998 (pp. 113–118). New York: Plenum Press, 1998

[10] *S. Thorpe, A. Delorme and R. VanRullen*, Spike-based strategies for rapid processing, in Neural Networks, **14**(6-7), 715-726, 2001