Methods of Learning for Spiking Neural Networks. A Survey

Conference Paper · October 2016

DOI: 10.1109/APEIE.2016.7806372

2 authors, including:

Andrey Gavrilov
Novosibirsk State Technical University
65 PUBLICATIONS **372** CITATIONS

SEE PROFILE

Methods of Learning for Spiking Neural Networks. A Survey

Andrey V.Gavrilov[1], Konstantin O.Panchenko[2]
[1]Novosibirsk State Technical University, Novosibirsk, Russia
[2]Motiv Ltd., Novosibirsk, Russia

***Abstract* – The paper aims to give general performance on achievements and trends in techniques for training of spiking neural networks in context of its hardware implementation, i.e. of neuromorphic technology.**

***Index Terms* – Spiking neural networks, training of neural networks, machine learning.**

I. INTRODUCTION

THE LAST decade is illuminated by rise of third generation of neural network – spiking neural networks (SNN). This boom of such neural networks is explained by appearance of opportunity to build large scale hardware implementation for them, which allows to speak about replacement of traditional von Neumann architecture by low-energy neural network based architecture of computers [1] in many fields. Such replacement demands answers on many questions:

- what kind (or kinds) of learning must be basic,
- how to learn SNN: offline in host computer previously or in real time during exploitation of SNN-chip,
- if in real time, that how this learning may be implemented in hardware (in CMOS, FPGA or other technologies),
- how to provide scalability and integration of neural system based super-computer to process Big Data or brain-like system,
- how to connect chips with SNN with sensors and actuators,
- how to code input and output data for SNN.

This paper focuses on discussion dealing with mostly first three questions.

It is possible follow approaches to select methods of learning for SNN:

- to try to translate well known methods and algorithms from classical artificial neural networks to SNN, e.g. error back propagation,
- to create novel methods and algorithms with respect to features of SNN.

In this paper examples of follow kinds of training methods for SNN are described:

- spike timing dependent plasticity (modified Hebb's rule),
- error back propagation,
- supervised Hebbian learning (SHL),
- remote supervision (method ReSuMe),
- growth evolving spiking neural networks,
- deep learning.

II. SPIKE TIMING DEPENDENT PLASTICITY

As known, proposed by Hebb since in 1940th the learning rule is based on neuro-phisiological research and read that efficacy of connections between neurons with similar states are increasing and otherwise are decreasing. In spike timing dependent plasticity (STDP) the change in efficacy of a synaptic weight depends of the time difference between the pre-synaptic and post-synaptic spikes [2, 3] (Fig. 1), i.e. supposed correlation between input spike and output spike of neuron.

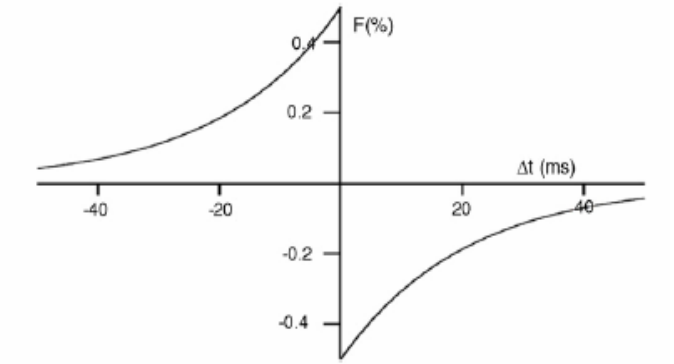

Fig. 1. Illustration of STDP learning rule [3]. Change of weight F depends on difference between pre-synaptic and post-synaptic spike times.

In [4] the recognition of digits from MNIST dataset based on SNN with STDP was described. The intensity values of the 28x28 pixel MNIST image are converted to Poisson-distributed spike trains with firing rates proportional to the intensity of the corresponding pixel. Those spike trains are fed as input to excitatory neurons in an all-to-all fashion. Excitatory neurons are connected to inhibitory neurons via one-to-one connections. Each inhibitory neuron is connected to all excitatory neurons, except for the one it receives a connection from. Class labels are not presented to the network, so the learning is unsupervised. Excitatory neurons are assigned to classes after training, based on their highest average response to a digit class over the training set. No additional parameters are used to predict the class, specifically no linear classifier or similar methods are on top of the SNN.

Besides the synaptic weight each synapse keeps track of another value, namely the presynaptic trace x_{pre}, which models the recent presynaptic spike history. Every time a presynaptic spike arrives at the synapse, the trace is increased by 1, otherwise x_{pre} decays exponentially. When a postsynaptic spike arrives at the synapse the weight change Δw is calculated based on the presynaptic trace

$$\Delta w = \eta(x_{pre} - x_{tar})(w_{\max} - w)^{\mu},$$

where:

η is the learning-rate,

w_{max} is the maximum weight,

μ determines the dependence of the update on the previous weight,

x_{tar} is the target value of the presynaptic trace at the moment of a postsynaptic spike. The higher the target value, the lower the synaptic weight will be.

This offset ensures that presynaptic neurons that rarely lead to firing of the postsynaptic neuron will become more and more disconnected and is especially useful if the postsynaptic neuron is only rarely active. A similar effect can be achieved by adding some noise to the input and adding a weight decrease mechanism to the learning rule (like in classical STDP in [2]).

In this approach the labels of classes are determined only after training selecting output neurons with maximum activity at presentation to neural network of corresponding image.

In [5] method STDP was employed to indirect control of mobile robot (virtual insect) moving to goal among possible obstacles.

The indirect spike-based training approach is illustrated on a seven-node SNN, shown in Fig. 2, that includes both excitatory and inhibitory synapses, n = 4 input neurons that receive information from each of the insect antennas, and r = 2 output neurons that control the two motors of the insect. Authors employ the rate coding to decode the output spikes of the neurons and Radial Basis Function spike model [6].

The simulations were conducted in MATLAB and in order to create the virtual environment, a 600x600 pixels image of the terrain and the target were generated. Simulations aimed to test the effectiveness of proposed training approach by comparing three trained states of the SNN including naive, partially-trained and fully trained on blank (a simple environment where the terrain has uniform smoothness and, therefore only target information is relevant), s-maze, and cloud terrains (most difficult task including often obstacles and narrow paths).

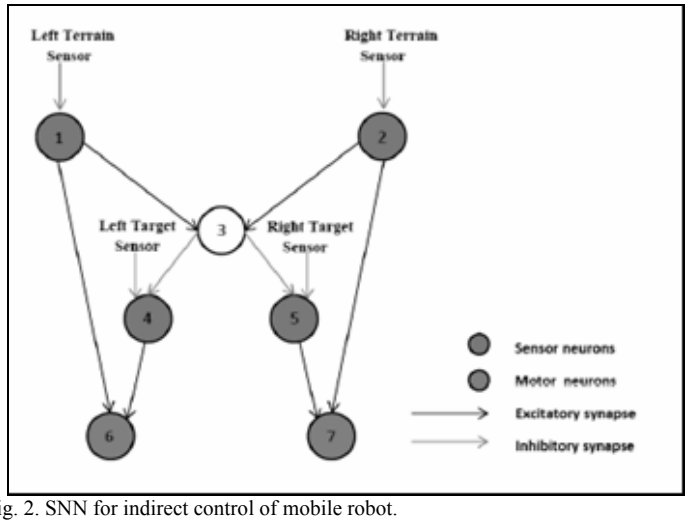

Fig. 2. SNN for indirect control of mobile robot.

Obtained results of simulation demonstrated employability of indirect control with STDP to solve navigation task of mobile robots.

Disadvantage of STDP is that it is usable only for unsupervised learning. Although there are attempts to combine this method with error back propagation (see below).

III. ERROR BACK PROPAGATION

Most important problem of supervised learning for spiking neural networks is binary character of inputs and outputs of neurons of such neural networks. So majority of known learning algorithms of conventional neural networks can't be directly employed as based on gradient ascent/descent techniques.

Nevertheless, in begin of 2000th researchers successfully tried to employ well-known method of error back propagation to train spiking neural networks [7]. Proposed in [7] learning algorithm was called as SpikeProp. Investigators used encoding of continues-variables for input and outputs of neurons by time of spike. So difference between desirable and actual time of output spike is employed as error. Thus in this approach single spike is interpreted as event and any modulation is absent.

SpikeProp algorithm has been re-investigated in [8, 9, 10, 11]. It was found that the weight initialization is a critical factor for a good performance of the learning rule. Moore [8] provided evidence that the training with large learning rates and with negative weights could be allowed while still leading to the successful convergence. These results were in contradiction to the conclusions of Bohte. Xin and Embrechts [10] proposed a modification of the learning algorithm by including the momentum term in the weight update equation. It has been demonstrated that this modification significantly speeded up the convergence of SpikeProp. In [9], Schrauwen and Campenhout adapted the gradient descent method derived in SpikeProp to adjust not only the synaptic weights, but also the synaptic delays, the time constants and the neurons' thresholds. This resulted in the faster algorithm convergence and in the smaller network topologies required for the given learning task. Finally, Tiňo and Mills [11] extended SpikeProp to recurrent network topologies to account for the temporal dependencies in the input stream. Neither the original SpikeProp method nor any of the discussed modifications enable learning of patterns composed of more than one spike per neuron.

This algorithm SpikeProp was verified in learning for classification using Iris dataset and enough good results (no worse than with Matlab BP and Matlab LM) was obtained.

Disadvantage of encoding in SpikeProp method is relatively difficult interpretation of results of classification, i.e. conversion of time interval to another representation more appropriate for using, for example, in control system of robot. Besides, it is difficult to prepare input data for processing by neural network. And assumption of single firings limits the class of neural information coding schemes implementable in the SpikeProp method. Moreover, the main drawback of the SpikeProp method is that there is no mechanism to "prop-up" the synaptic weights once the postsynaptic neuron no longer fires for any input pattern.

Paper [12] consists of a good comparison of supervised learning methods based on time-spiking encoding.

Another approach using error back propagation was employed recently to train neural network deploying in neuro-chip TrueNorth of IBM [13]. In this case authors use offline training and offer to replace input and output spikes by their probabilities. It causes to employ EBP.

Thus, training follows the back propagation methodology by iteratively:

1) running a forward pass from the first layer to the last layer,
2) comparing the network output to desired output using a loss function,
3) propagating the loss backwards through the network to determine the loss gradient at each synapse and bias term,
4) using this gradient to update the network parameters.

The training network forward pass is a probabilistic representation of the deployment network forward pass. Synaptic

connections are represented as probabilities $\tilde{c}_{ij}$, where $P(c_{ij} = 1) = \tilde{c}_{ij}$, while synaptic strength is represented using s_{ij} as in the deployment network. It is assumed that s_{ij} can be drawn from a limited set of values and we consider the additional constraint that it is set in "blocks" such that multiple synapses share the same value, as done in TrueNorth for efficiency. While it is conceivable to learn optimal values for s_{ij} under such conditions, this requires stepwise changes between allowed values and optimization that is not local to each synapse. Authors take a simpler approach, which is to learn biases and synapse connection probabilities, and to intelligently fix the synapse strengths using special optimization task minimizing redundancy of neurons during initializing of network in chip. Input to the training network is represented using $\tilde{x}_{ij}$, which is the probability of an input spike occurring in the deployment network.

After learning the neural network with obtained weights (strengths) of synapses and probabilities of connections between neurons in core is deploying into neuro-chip with respect to features of structure of SNN implemented there (binary input and output signals, limits of parameters representation and so on). At that every recognizing class (output neuron) is provided by ensemble of several neural networks in which matrix of synaptic connections are generated randomly according with averages (real values) obtained during training.

This matrix from binary values determines existing connections between *i*-th synapse and *j*-th neuron in core of TrueNorth. Summed signal for *j*-th neuron is calculated as:

$$I_j = \sum_i x_i c_{ij} s_{ij} + b_j \qquad (2)$$

where:

x_i – input signal in *i*-th synapse,

s_{ij} – weight (strenth) of *i*-th synapse for *j*-th neuron,

b_j – bias of *j*-th neuron.

Note that authors does not change weights during learning of neural network but change only probabilities of synaptic connections. Веса задаются уже в процессе размещения нейронной сети в чипе, как результат решения оптимизационной задачи минимизации избыточности нейронов.

This method was verified by learning for well known task of classification of Iris. Authors had obtained enough good results for number of neural networks in ensemble 1, 4, 16 and 64 from 92.7% to 99.42% accuracy correspondingly.

IV. SUPERVISED HEBBIAN LEARNING

Supervised Hebbian Learning (SHL) [28] is arguably the most biologically plausible supervised SNN learning algorithm. SHL simply seeks to ensure that an output neuron fires at the desired time, with the inclusion of a 'teaching' signal. Since the teaching signal comprises of intracellular synaptic currents, supervision may be envisioned as supervision by other neurons. Method SHL does suffer from the limitation that even after the goal firing pattern has been reached; SHL continues to change the weights. Thus, constraints must be added to the learning rule to ensure stability. However, the problem with setting constraints is that it is not easy to know at which point in the training they should be applied. The weights will continue to increase after each training epoch and eventually could cause the network to be unstable, or at least to generalize poorly in the testing phase of learning [12].

Authors in [28] proposed one of the first spike-based methods similar to SHL approach. In the first attempt, they defined the learning rule for the monosynaptic excitation. The learning process was based on three spikes (two pre-synaptic and one post-synaptic) generated during each learning cycle. The first presynaptic spike at the time t_1^{in} was considered as in input signal, whereas the second presynaptic spike at $t_2^{in}=t^d$ pointed to the target firing time for the postsynaptic neuron. The learning rule is

$$\Delta w = \eta(t^{out} - t^d),$$

where $\eta > 0$ is the learning rate and t^{out} is the actual time of the postsynaptic spike.

In [29] authors proposed modified Supervised Hebbian Learning based on concept of Fuzzy Spiking Neural Network (FSNN) and encoding by frequency modulation.

The input neuron is responsible for simply encoding the feature data into an appropriate frequency range. The spike trains are generated by linear encoding scheme. The encoding scheme takes the frequency data points and converts them into an inter-spike-interval (ISI) which is used to create input spike train. Each data point scaled into a particular frequency range in linearly scaled into an input spike train of a particular sample length.

Gaussian RFs are placed at every synapse between the input and hidden neurons. The job of RFs is to determine the relation between the input frequencies fi and the central operating frequency F0 of RF. The weight is then scaled by ki. The process relates to the IF (xi is Ai) part of fuzzy rule, where xi is input and Ai represent the RF. By the use of RFs, proper tuning of dynamic synapse is not required. The function of each hidden layer neuron is to impose the remaining part of antecedent fuzzy IF-THEN rule, the conjunctive 'AND', now summing the PSP by performing the disjunctive 'OR'. The main function of RFs connecting to hidden layer neuron is to filter the spikes to the output layer.

The action potential with the synapse is only significantly high in magnitude for a very short interval of time. This type of synapse has been considered as coincidence detector. It is then the task of supervised learning algorithm to associate the hidden layer neurons to the output layer neurons, thus performing fuzzy reasoning between the hidden layer (antecedents), and the output layer (consequents).

STDP learning window is used to modify the synaptic efficacy between the hidden and output neurons. This modified form of SHL does not require an actual supervisory spike train as with the [28] approach.

This approach was implemented in MATLAB.

V. REMOTE SUPERVISION

In [14, 15] the learning method ReSuMe was suggested based on multiple spike. The goal of the ReSuMe learning is to impose on a neural network the desired input-output properties, i.e. to produce the desired spike trains in response to the given input sequences.

ReSuMe takes advantage of the Hebbian (correlation) processes and integrates them with a concept of remote supervision. The name 'remote supervision' comes from the fact that the target signals are not directly delivered to the learning neurons (as it is the case in SHL), however they still co-determine the changes of the synaptic efficacies in the connections terminating at the learning neurons. In more details,

a synaptic efficacy w_{ki}, between any given presynaptic neuron $n^{in}_k(i)$ and a corresponding postsynaptic neuron n^{out}_i , is modified according to two rules. The first rule depends on the correlation between $n^{in}_k(i)$ and n^{out}_i firing times. The second rule is determined by the correlation between $n^{in}_k(i)$ and $n^{d}_j(i)$ firing times. By $n^{d}_j(i)$ we denote a 'teacher' neuron delivering the target signal for n^{out}_i . For the excitatory synapses these two rules have the forms similar to STDP and anti-STDP and are described by the learning windows $W^d(s^d)$ and $W^{out}(s^{out})$ (Fig. 3).

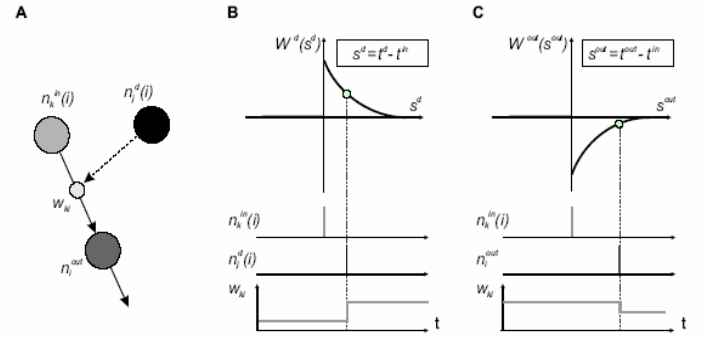

Fig. 3. Concept of remote supervision and learning windows in ReSuMe.

Method ReSuMe was verified by training of SNN for reconstruction of the spatio-temporal patterns of spikes corresponding to the patterns of activity in the pools of motor neurons. Each pool, consisting of 40 neurons, activated the particular muscle model. The simplified limb model, driven by the SNN, was able to follow the desired trajectory of movement with a satisfactory precision.

VI. GROWTH EVOLVING SPIKING NEURAL NETWORKS

Some investigators are working under training of SNN based on evolution programming (or genetic algorithms). This approach was proposed in [18] for learning of SNN with limited codes of weights and delays of synapses (3 bits). Authors employed spike-time coding as output data and mean-squared error as fitness function. This approach was verified by task of classification of Fisher iris. Enough good results was obtained and comparison its with similar results obtained by usage of different modifications of SpikeProp. However, in [18] authors did not use opportunity to employ genetic algorithms to build growth spiking neural networks (with creation of new neurons and connections).

In [19] gene-driven network growth algorithm that enables a genetic algorithm (evolutionary computation) to generate and test SNNs was suggested. The genome not only specified the network topology, but all its parameters as well. In experiments, the algorithm discovered SNNs that effectively produce a robust spike bursting behavior given tonic inputs, an application suitable for central pattern generators where the spike bursts activate the walking or swimming behavior of some robot. Even though evolution did not include perturbations of the input spike trains, the evolved networks showed remarkable robustness to such perturbations. On a second task, a sequence detector, several related discriminating designs were found, all made "errors" in that they fired when input spikes were simultaneous (i.e. not strictly in sequence), but not when they were out of sequence. They also fired when the sequence was too close for the teacher to have declared they were in sequence. That is, evolution produced these behaviors even though it was not explicitly rewarded for doing so.

Topology is represented by matrix of connections and is encoding unto chromosome by special algorithm. Except topology the chromosome includes information about following parameters for each neuron: absolute refractory period (during which no amount of excitation can make it fire), time constant controlling exponential return of membrane potential to resting value from depolarized value.

Besides, chromosome includes for each synapse: weight, delay, time constants defining exponential rise and fall of PSP (post synaptic potential).

Similar approach was proposed in [20]. In this paper authors use less number of parameters of neurons to train neural network and train SNN to solve task of classification. Proposed algorithms were verified in tasks of classification of iris, diagnostics of breast cancer, hepatitis, heart and ionosphere.

Disadvantage, common to all evolutionary algorithms, is that the computation with this approach is very time consuming. Advantage is opportunity to use for evaluating practically either parameter of neural network and neurons.

In [21] authors offer another approach to growth spiking neural networks without use of genetic algorithms. The authors offer novel class of SNN dynamic evolving SNN (deSNN) oriented to on-line learning and recognition of spatio- and spectro-temporal data. This approach is based on concept of evolving spiking neural network proposed in [21] and some similar to Adaptive Resonance Theory [22] for conventional neural networks. In this approach authors use Euclidean difference between input vector and center of cluster to decide creating of new output neuron (cluster) or no. In last case it is considered that input pattern is recognized.

The authors in edSNN utilize rank-order learning and Spike Driven Synaptic Plasticity SDSP (variant of STDP) spike-time learning in unsupervised-, supervised-, or semi-supervised modes. The SDSP learning is used to evolve dynamically the network changing connection weights that capture spatio-temporal spike data clusters both during training and during recall. The model deSNN was verified on two case study applications: (1) moving object recognition using address-event representation (AER) with data collected using a silicon retina device; (2) EEG SSTD recognition for brain-computer interfaces.

VII. DEEP LEARNING

Deep-learning neural networks such as convolutional neural network (CNN) have shown great potential as a solution for difficult vision problems, such as object recognition.

In [24, 25, 26] the approach based on conversion of convolution neural network to spiking neural network is suggested.

In [24] an approach for converting a deep CNN into a SNN is proposed, that enables mapping CNN to spike-based hardware architectures. This approach first tailors the CNN architecture to fit the requirements of SNN, then trains the tailored CNN in the same way as one would with CNN, and finally applies the learned network weights to an SNN architecture derived from the tailored CNN. Authors evaluated the resulting SNN on publicly available Defense Advanced Research Projects Agency (DARPA) Neovision2 Tower and CIFAR-10 datasets and show similar object recognition accuracy as the original CNN. Obtained SNN implementation is amenable to direct mapping to spike-based neuromorphic hardware, such as the ones being

developed under the DARPA SyNAPSE program (chip TrueNorth). Hardware mapping analysis of authors suggests that SNN implementation on such spike-based hardware is two orders of magnitude more energy-efficient than the original CNN implementation on off-the-shelf FPGA-based hardware.

In [25] authors train spiking deep networks using leaky integrate-and-fire (LIF) neurons, and achieve state-of-the-art results for spiking networks on the CIFAR-10 and MNIST datasets. They achieved this result by softening the LIF response function, such that its derivative remains bounded, and by training the network with noise to provide robustness against the variability introduced by spikes.

After training the trained static network is converting to a dynamic spiking network. The parameters in the spiking network (i.e. weights and biases) are all identical to that of the static network. The convolution operation also remains the same, since convolution can be rewritten as simple connection weights (synapses) w_{ij} between pre-synaptic neuron i and post-synaptic neuron j. Similarly, the average pooling operation can be written as a simple connection weight matrix, and this matrix can be multiplied by the convolutional weight matrix of the following layer to get direct connection weights between neurons.

This method is general and could be applied to other neuron types, including those used on modern neuromorphic hardware.

In [26] authors analyze the effects of converting deep ANNs into SNNs with respect to the choice of parameters for spiking neurons such as firing rates and thresholds. Authors present a set of optimization techniques to minimize performance loss in the conversion process for ConvNets and fully connected deep networks. These techniques yield networks that outperform all previous SNNs on the MNIST database to date, and many networks here are close to maximum performance after only 20 ms of simulated time. The techniques include using rectified linear units (ReLUs) with zero bias during training, and using a new weight normalization method to help regulate firing rates.

This method for converting an ANN into SNN enables lowlatency classification with high accuracies already after the first output spike, and compared with previous SNN approaches it yields improved performance without increased training time.

In [27] authors developed a scheme for learning connectivity in a deep spiking neural network. The scheme is based on learning conditional instantaneous firing rates, linking it to many of the statistical frameworks previously developed in deep learning that are based on conditional probabilities.

An event based dataset of moving MNIST digits collected using a DVS camera was used to verify proposed scheme of learning of SNN for both prediction and classification tasks.

VIII. CONCLUSION

Thus, there are many different approaches to train spiking neural networks. Most of them are Spike Timing Dependent Plasticity (STDP) and its modifications and Error Back Propagation (EBP) for event-driven model of spiking neural networks (for single output spike based learning). Actually another encoding schemes are not employing. Although it seems that frequency modulation may be useful to expand functionality and effectiveness of spiking neural networks. Conversion of usual performance of neural network dealing with continuous values to event-driven hardware implemented neural networks often demands great superfluity of neurons and connections.

So we think that most important problems of learning for spiking neural networks to widely use them are following:

- development of formalized, effective and universal methods of conversion of trained conventional neural networks to deployment spiking neural network,
- development of growth spiking neural networks without genetic algorithms similar to Adaptive Resonance Theory (ART) for conventional neural networks,
- development of more than EBP bio-inspired approach to train SNN for supervised learning,
- investigation of opportunities to build hardware implemented real time learnable spiking neural networks.

REFERENCES

[1] Andrew S. Cassidy a,1, Julius Georgiou b, Andreas G. Andreou. Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization // Neural Networks, 2013, № 45, pp. 4–26.

[2] Bi,G.-Q., Poo,M. (1998). Synaptic modifications in cultured hippocampal neurons :dependence on spike timing, synaptic strength, and postsynaptic cell type // J.Neurosci., 1998б 18, pp. 10464–10472.

[3] Song S, Miller KD, Abbott LF. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity // Nat Neurosci., 2000, 3(9), :pp. 919-926.

[4] Peter U.Diehl, Matthew Cook..Unsupervised learning of digit recognition using spike-timing-dependentplasticity // Frontiers in Computational Neuroscience, August 2015, Volume 9, Article 99.

[5] Zhang X., Xu_ Z., Henriquez C., Ferrari S. Spike-Based Indirect Training of a Spiking Neural Network-Controlled Virtual Insect // Proc. of 52nd IEEE Conference on Decision and Control December 10-13, 2013. Florence, Italy.

[6] Zhang X., Foderaro G., Henriquez C., VanDongen A. M. J., Ferrari S., A radial basis function spike model for indirect learning via integrate-and-fire sampling and reconstruction techniques // Advances in Artificial Neural Systems, Volume 2012 (2012), Article ID 713581.-http://dx.doi.org/10.1155/2012/713581

[7] Sander M. Bohtea; Joost N. Koka, Han La Poutr. Error-backpropagation in temporally encoded networks of spiking neurons // Neurocomputing, 48, 2002, pp. 17–37.

[8] Moore S. C.. Back-Propagation in Spiking Neural Net-works. MSc thesis, University of Bath, 2002.

[9] Schrauwen B., Campenhout J. V. Improving SpikeProp: Enhancements to an Error-Backpropagation Rule for Spiking Neural Networks // Proceedings of the 15th ProRISC Workshop, 11 2004.

[10] Xin J., Embrechts M. J. Supervised Learning with Spiking Neuron Networks // Proceedings IEEE International Joint Conference on Neural Networks, IJCNN'01, Washington D.C., July 15-19 2001, pp. 1772–1777.

[11] Tiňo P. Mills A. J. Learning Beyond Finite Memory in Recurrent Networks of Spiking Neurons. In L. Wang, K. Chen, and Y. Ong, editors, Advances in Natural Computation // ICNC 2005, Lecture Notes in Computer Science, Springer-Verlag, 2005, pp. 666–675.

[12] Kasinski A., Ponulak F. Comparison of Supervised Learning Methods for Spike Time Coding in Spiking Neural Networks // Int. J. Appl. Math. Comput. Sci., 2006, Vol. 16, No. 1, pp. 101–113.

[13] Esser S.K, Appuswamy R., Merolla P.A., Arthur J.V., Modha D.S. Backpropagation for Energy-Efficient Neuromorphic Computing // Proc. of Int. Conf. Neural Information Processing Systems 28 (NIPS 2015), Montreal, Canada, 2015.

[14] Ponulak F., Kasiński A., Supervised learning in spiking neural networks with ReSuMe sequence learning, classification, and spike shifting // Neural Comput., 2010, vol. 22, pp. 467-510.

[15] Ponulak F.. Supervised learning in Spiking Neural Networks with ReSuMe Method. Doctoral Dissertation. Poznań University of Technology, 2006.

[16] Carnell A., Richardson D. Linear Algebra for Time Series of Spikes, Proc. of ESANN'2005 - European Symposium on Artificial Neural Networks Bruges (Belgium), 27-29 April 2005, pp. 363-368.

[17] Cohen H.. A Course in Computational Algebraic Number Theory. - Springer-Verlag, New York, 1993.

[18] Belatreche A., Maguire L. P., McGinnity M., Wu Q. X. A Method for Supervised Training of Spiking Neural Networks. // Proceedings of IEEE Cybernetics Intelligence - Challenges and Advances (CICA), 2003, Reading, UK, pp. 39–44.

[19] Schaffer J. D.. Evolving spiking neural networks: A novel growth algorithm corrects the teacher // Proc. of 2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). 26-28 May 2015, pp. 1-8.

[20] Abdulrazak Yahya Saleh, Haza Nuzly Bin Abdull Hameed, Mohd Najib Mohd Salleh. A Novel hybrid algorithm of Differential evolution with

Evolving Spiking Neural Network for pre-synaptic neurons Optimization // Int. J. Advance Soft Compu. Appl, Vol. 6, No. 1, March 2014.
[21] Kasabov N., Dhoble K., Nuntalid N., Indiveri G. Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition // Neural Networks, 41, 2013, pp. 188–201.
[22] Wysoski, S., Benuskova L., Kasabov N. On-line learning with structural adaptation in a network of spiking neuro ns for visual pattern recognition // Artificial Neural Networks–ICANN 2006, 2006, pp. 61-70.
[23] Carpenter G., A., Grossberg S. Pattern Recognition by Self-Organizing Neural Networks. - Cambridge, MA, MIT Press, 1991.
[24] Yongqiang Cao, Yang Chen, Deepak Khosla. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition // Proc, of International Journal of Computer Vision. May 2015, Volume 113, Issue 1, pp 54-66.
[25] Hunsberger E., Eliasmith C. Spiking Deep Networks with LIF Neurons. - arXiv:1510.08829, 2015.
[26] Diehl P.U. Neil D., Binas, J., Cook, M., Liu, S.C., Pfeiffer, M. Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing // Proc. of IEEE International Joint Conference on Neural Networks (IJCNN), 2015.
[27] J. A. Henderson, T. A. Gibson, J. Wiles. Spike Event Based Learning in Neural Networks. - arXiv:1502.05777, 2015.
[28] Ruf, B., Schmitt, M. Learning temporally encoded patterns in networks of spiking neurons // Neural Processing Letters, 5(1), 1997, pp. 9-18.
[29] Rasa H., Tsegaye, Biswas R. Fuzzy based modified SHL algorithm for spiking neural networks // Int. Journal of Computer Applications, vol. 41, N. 5, March, 2012, pp. 33-37.

Andrey V. Gavrilov received the System Engineer (Automatic Control Systems) degree from Novosibirsk Electric Technical Institute. He is presently working as Associate Professor (Docent) in Novosibirsk State Technical University and has 35+ years of teaching experience to undergraduate and graduate students, is member of Program Committee of several International conferences, editor of CEJCS, has more 140 publications. His areas of interests are hybrid intelligent systems, neural networks, natural language processing, ambient intelligence, cognitive robotics.

Konstantin E. Panchenko graduated from Novosibirsk State Technical University (computer engineering and automated systems) in 2008. Has 8+ years of experience in software development projects. His areas of interests are neuromorphic technologies, neural networks, technical vision.