ISSN 0027-1349, Moscow University Physics Bulletin, 2024, Vol. 79, Suppl. 2, pp. S944–S952. © Allerton Press, Inc., 2024.

Spiking Neural Network Actor–Critic Reinforcement Learning with Temporal Coding and Reward-Modulated Plasticity

D. S. Vlasov[1*], R. B. Rybka[1], A. V. Serenko[1], and A. G. Sboev[1,2]

[1] *National Research Centre "Kurchatov Institute," Moscow, Russia*
[2] *National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russia*

Received October 1, 2024; revised October 10, 2024; accepted October 20, 2024

Abstract—The article presents an algorithm for adjusting the weights of the spike neural network of the actor–critic architecture. A feature of the algorithm is the use of time coding of input data. The critic neuron is applied to calculate the change in the expected value of the action performed based on the difference in spike times received by the critic when processing the previous and current states. The change in the weights of the synaptic connections of the actor and critic neurons is carried out under the influence of local plasticity (spike–timing-dependent plasticity), in which the change in weight depends on the received value of the expected reward. The proposed learning algorithm was tested to solve the problem of holding a cart pole, in which it demonstrated its effectiveness. The proposed algorithm is an important step towards the implementation of reinforcement learning algorithms for spiking neural networks on neuromorphic computing devices.

Keywords: spiking neural networks, reinforcement learning, temporal coding, stdp, r-stdp

DOI: 10.3103/S0027134924702400

1. INTRODUCTION

Recently, a number of studies of energy-efficient devices have appeared, including hardware neural networks [1–3], which can give an increase in energy efficiency relative to GPUs and TPUs [4]. Modern neuroprocessors consuming milliwats of energy [5], such as TrueNorth [6], Loihi [7], Tianjic [8], Altai[1] are capable of performing various tasks related to image and audio data analysis, as well as managing agents.

Using hardware implementations of spiking neural networks (SNNs), in which information is represented in the form of electronic pulses-spikes [9], may provide the most promising gains in energy efficiency [5]. Such implementations can be based on memristive devices, in which the equivalent of the synaptic weight is the memristor conductance. It combines the functions of information processing and storage [10, 11] and allows for fast and efficient vector matrix multiplication operations [12–14]. However, using the backpropagation method [15] to train spiking neural networks implemented on meristive devices is problematic, since computing each weight update requires information about almost the entire system [16]. On the other hand, using learning algorithms based on local synaptic plasticity will allow each synapse to learn independently, since only information from pre- and postsynaptic neurons is required to change the weight [17]. This allows using memristor-based hardware not only for inference of the trained network, but also for its training [18, 19].

Memristive devices support a model of local plasticity called spike-time dependent plasticity (STDP) [20], in which the change in synaptic weight is determined by the relative timing of presynaptic and postsynaptic spikes (Fig. 1). It has been shown that various types of memristors exhibit plasticity similar to STDP [12, 17, 21, 22] and the change in the conductivity of the memristor depends on the temporary overlap of presynaptic and postsynaptic spikes. There are various methods for training SNN with memristive STDP to solve image and vector classification problems [23–26].

In reinforcement learning (RL) agent learns to interact with the external environment by receiving rewards or punishments for its actions. This type of learning is biologically plausible and its implementation on memristive devices makes it possible to

* E-mail: vfked0d@gmail.com

[1] https://motivnt.ru/neurochip-altai.

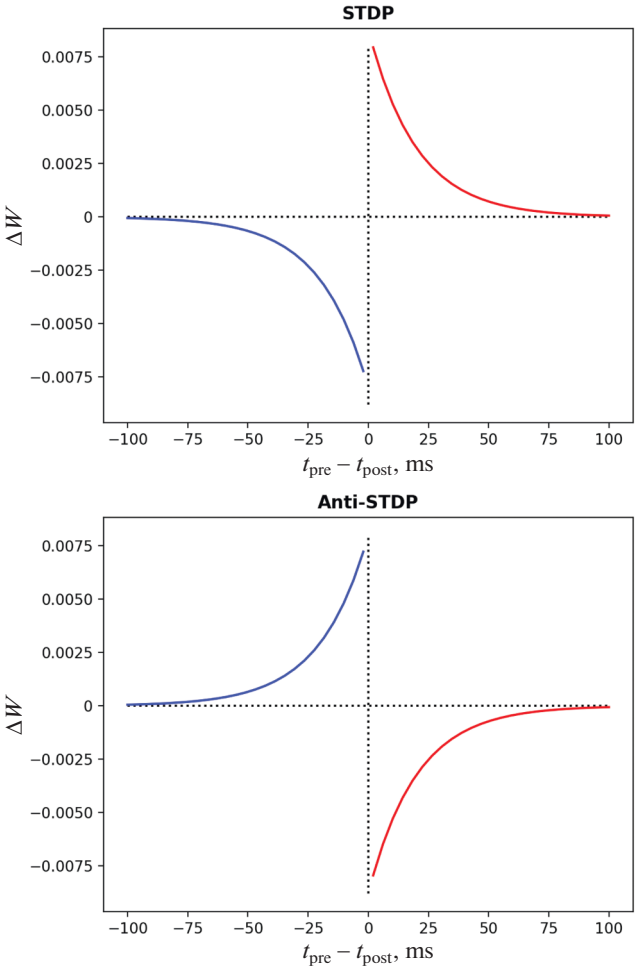

Fig. 1. Spike-time dependent plasticity.

solve control problems on autonomous robotic devices. Most of the RL algorithms existing in [27–29] are based on time-difference learning [30]. This methods are based on difference of reward expected by critic neuron and effectively solve benchmark tasks like CartPole, Acrobot. However, they are not intended for implementation in memristive devices.

Frequency coding is most often used in these works. However, more modern encoding methods are known in [31], including temporal coding. It involves a smaller number of spikes and allow obtaining better signal-to-noise characteristics. Thus, in [32] it was shown that a spiking network with temporal coding is able to solve the cart-pole problem by reinforcing neurons with pulses of external current.

In this paper, a network with temporal coding is extended by a neuron critic and learning is performed by modulating synantic plasticity depending on the reward received from the critic. This is significant step for the future implementation of this RL algorithms for SNN on neuromorphic computing devices.

Next, the Subsection 2.5 describes the CartPole task that was used for testing the proposed RL algorithm (see Subsection 2.6). The architecture of spiking neural network as a part of the RL algorithm and its components are described in Subsections 2.1, 2.2, and 2.4. The Subsection 2.3 shows a method for converting environmental states into spike moments. The experiments and results are discussed in the Section 3.

2. METHODS AND APPROACHES

2.1. Neuron Model

In the proposed network, the dynamics of neurons are described by leaky integrate-and-fire (LIF) model of spiking neuron. We preferred it to more complex and more biologically plausible models, such as the Fitzhugh–Nagumo model [33], because the LIF neuron can be prospectively implemented in hardware based on memristors [34], and at the same time it has been shown to be suitable for supervised learning methods for SNN [35, 36], including using time coding [32, 37, 38].

In the LIF neuron, the membrane potential $V(t)$ obeys

$$\frac{dV}{dt} = -\frac{V(t) - V_{\text{rest}}}{\tau_{\text{m}}} + \frac{I_{\text{syn}}(t) + I_{\text{ext}}(t)}{C_{\text{m}}}.$$

When $V(t)$ exceeds the threshold value of V_{th}, the neuron emits a spike signal. Subsequently, V is reset to $V_{\text{rest}} = 0$, and during the refractoriness period t_{ref} the neuron is insensitive to input spikes. The membrane relaxation constant τ_{m} corresponds to time, while the membrane potential $V(t)$ decreases to V_{rest} when the neuron is not stimulated enough to receive an output spike. The neuron membrane capacity C_{m} was equal 1 pF.

$I_{\text{ext}}(t)$ is an external stimulation current. $I_{\text{syn}}(t)$ is the incoming postsynaptic current entering the ith input synapse at time t^i_{sp} depending on the synaptic weight $w_i(t)$:

$$I_{\text{syn}}(t) = \sum_i \sum_{t^i_{\text{sp}}} w_i(t) \frac{q_{\text{syn}}}{\tau_{\text{syn}}} e^{-\frac{t - t^i_{\text{sp}}}{\tau_{\text{syn}}}} \Theta\left(t - t^i_{\text{sp}}\right).$$

Here $q_{\text{syn}} = 5$ fC is the magnitude of the electric charge passing through the synapse, and $\tau_{\text{syn}} = 5$ ms is the time constant of the synaptic ion channels. The synaptic weight $w_i(t)$ changes after each presynaptic and postsynaptic spike in accordance with the plasticity model.

The constants V_{th}, τ_{m}, and t_{ref} are adjusted according to the task. t_{ref} is chosen such that there is only one spike per input pattern.

2.2. STDP

In this paper, numerical experiments are performed with the usual STDP in its additive form [39], where the change in synaptic weight Δw does not depend on the current weight w, but depends only on the time interval $\Delta t = t_{\text{post}} - t_{\text{pre}}$, where t_{post} and t_{pre} are the times of appearance of post- and presynaptic spikes:

$$\Delta w = \begin{cases} -\lambda \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_-}\right), \\ \text{if} \quad t_{\text{post}} - t_{\text{pre}} < 0; \\ \lambda \exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_+}\right), \\ \text{if} \quad t_{\text{post}} - t_{\text{pre}} > 0. \end{cases} \tag{1}$$

At the same time for changing weights we applyed the anti-STDP, which is a mirror version of STDP:

$$\Delta w = \begin{cases} \lambda \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_-}\right), \\ \text{if} \quad t_{\text{post}} - t_{\text{pre}} < 0; \\ -\lambda \exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_+}\right), \\ \text{if} \quad t_{\text{post}} - t_{\text{pre}} > 0. \end{cases} \tag{2}$$

Figure 1 shows the curves of the dependence of the change in weight on the difference between t_{post} and t_{pre} for STDP (Eq. (1)) and antiSTDP (Eq. (2)).

2.3. Coding States to Spikes

In our research environment state vectors are transformed into suitable for SNNs spike sequences using time coding. The time coding representing input values by spike times was chosen to ensure the minimum number of spikes needed and the minimum power consumption.

Before transforming the feature vectors into spike sequences, two preprocessing steps are performed sequentially. First, the input data is normalized such that each component of the feature vector takes values between 0 and 1 using MinMaxScale.[2] The resulting feature vectors are then processed with Gaussian receptive fields (GRF) [40–42] increasing the dimensionality of the input data while ensuring that all vectors have the same amount of spikes per vector. The N-dimensional state vector of the environment x is transformed into a vector of dimension NM, where M is the number of receiving fields (Fig. 2).

Each component of the vector x^i is transformed into M components $g(x^i, \mu_0), \ldots, g(x^i, \mu_M)$, where

$$g(x^i, \mu_j) = \exp\left(\frac{(x^i - \mu_j)^2}{\sigma^2}\right),$$

[2] https://scikit-learn.org/stable/modules/preprocessing.html.

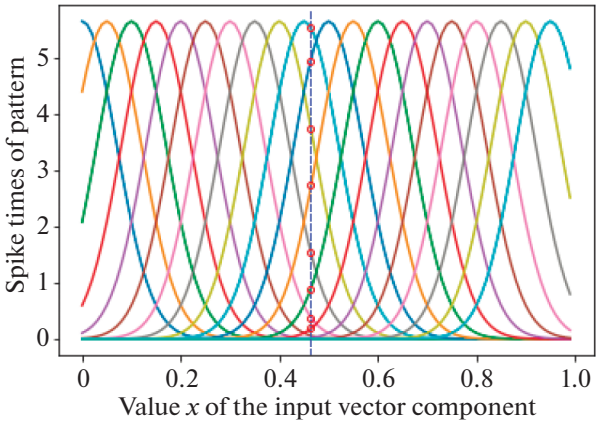

Fig. 2. Gaussian receptive fields coding.

where μ_j is the center of the jth receptive field:

$$\mu_j = X^i_{\min} + (X^i_{\max} - X^i_{\min})\frac{j}{M-1},$$

where $X^i_{\max}$ and $X^i_{\min}$ are the maximum and minimum values of the ith component among all vectors of the training set.

After preprocessing with receptive fields, the components g_i of the resulting vector are encoded by spike times: the ith input synapse of each action neuron receives an input spike at time $T(1 - g_i)$ relative to the start of the representation of the current state of the input environment. Thus, the maximum possible input value 1, reflecting the fact that the environmental state value is at the center of the corresponding receptive field, is encoded by an early spike, and low values of the pre-processed input vector are encoded by late spikes in the chosen spike feeding time window ($t_h = 50$ ms).

2.4. Topology

The network contains:

1. Input neurons equal to the length of the environment state vector multiplied by the number of receptive fields described in Subsection 2.3;

2. Output neurons responsible for actions that consists of LIF neurons described previously in Subsection 2.1;

3. Critic neuron responsible for calculating potential reward.

Input neurons receive spike moments obtained after transforming the environment state by receptive fields (see Subsection 2.3). Input neurons are connected to the output layer by synapses with synaptic plasticity

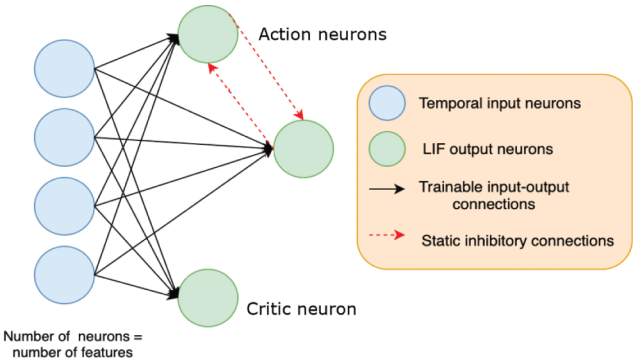

Fig. 3. Network consists of two action neurons interconnected with inhibitory connections and one critic neuron. All output neurons connected to input neurons via plastic connections. Input neurons receive temporal patterns having one spike per input neuron. Action corresponds to neuron emiting earliest spike. Value is defined by precise moment of output spike on critic neuron.

STDP described in Subsection 2.2. Neurons responsible for action are linked by nonplastic inhibitory connections to create competition in determining the action predicted by neurons responsible for action. The network topology is shown in Fig. 3.

2.5. The Cart-Pole Task Formulation

In the cart-pole problem a cart with a pole attached to it by a hinge (Fig. 4) can move along a surface in one dimension without friction. The agent can apply a horizontal force of 10 N to the cart, pushing it left or right. All physical parameters of the system are same as in [43].

The pole's initial position is vertical, and the agent should keep it from falling as long as possible. The episode fails when the pole deviates from the vertical by more than 12° or the cart moves more than 2.4 m from the center. The episode ends successfully when the maximum number of time steps $S_{\max} = 200$ for CartPole-v0 and $S_{\max} = 500$ for CartPole-v1 are reached. A reward of $+1$ is awarded for each time step during which the pole remains in an upright position.

The problem is considered solved if the cart control algorithm keeps the pole from falling for at least 195 time steps for the CartPole-v0 problem and 475 for CartPole-v1 problem on average over 100 episodes. At each step, the environment modeling the cart with a pole returns the values of the coordinate x and the velocity $\dot{x}$ of the cart, as well as the angle of deviation (θ) of the pole and its angular velocity ($\dot{\theta}$). Based on these four parameters, the agent have to choose the direction of application of force in the next time step: either to the left or to the right.

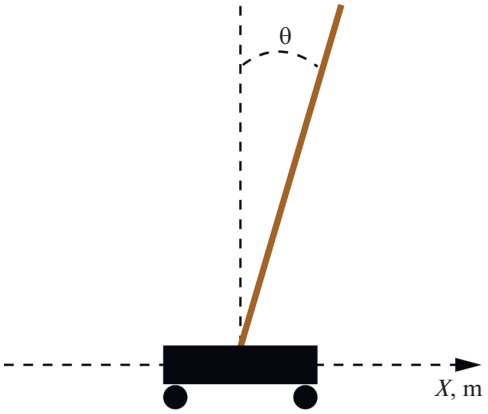

Fig. 4. The cart and the pole.

2.6. Learning Algorithm

The learning algorithm 1 is based on the difference in the predictions of the critic for current and previous states. Achieved value Δr affects on synaptic weight changing. Additionally, depending on the Δr sign, the plasticity switches from STDP to anti-STDP.

The learning algorithm works successfully only if the weight changes are applied only to the pair: the critic neuron and the neuron that performed the action. The coefficient λ from Eqs. (1) and (2) is calculated dynamically for each "i" episode based on the running reward R_i (Eq. (3)) and the number of actions played in the previous game S_i multiplied by the coefficient K (Eq. (4)) to lower learning rate and prevent weights from destabilizing. K is calculated for each version of CartPole task differently using Eq. (5) for v0 and Eq. (6) for v1:

$$R_{i+1} = 0.05S + 0.95R_i, \tag{3}$$

$$\lambda = \lambda_0 \Delta r K(S_i, R_{i+1}). \tag{4}$$

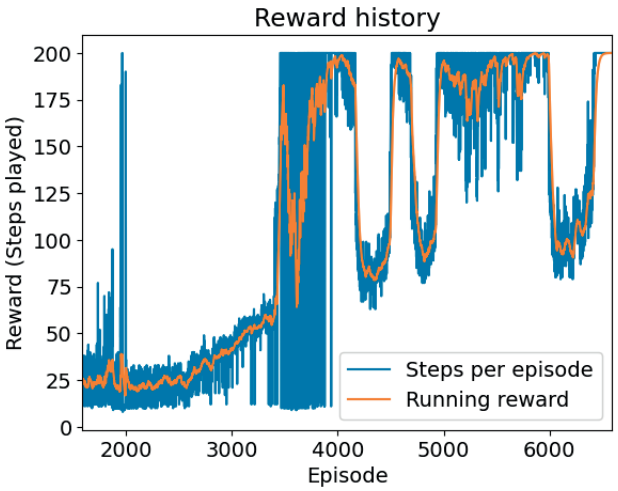

Fig. 5. Running reward and steps played per episode during learning of CartPole-v0 task.

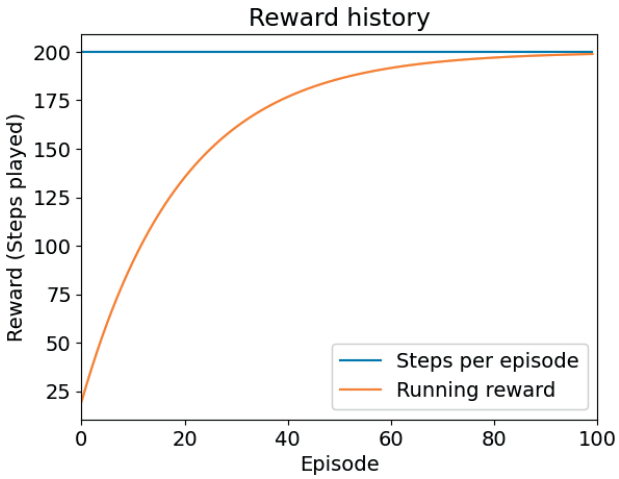

Fig. 6. Running reward of CartPole-v0 test simulation. Orange line shows that at the end of 100 episodes reward is higher than 195, so task is considered as solved.

If the network maintains equilibrium for $S_{\max}$ steps, then at the next iteration the weight change is disabled. Slowing down the weight change or disabling it altogether is necessary because the network can lose the equilibrium at which it solves the problem if the learning rate λ is high.

Another feature of the learning method is that in order to change the weight, the state of the environment has to be rerun through the network so that the weight changes. The decision-making time of the SNN model is about 50 ms, just as the time for changing the weight is the same 50 ms.

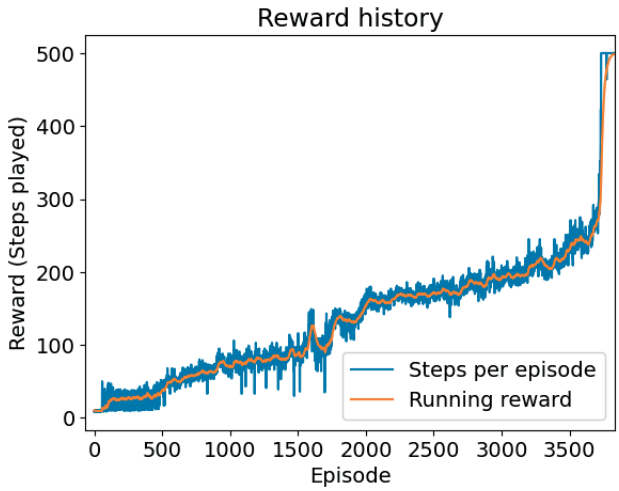

Fig. 7. Running reward and steps played per episode during learning of CartPole-v1 task.

3. RESULTS

In [32] it was shown that this network topology is capable of solving the CartPole-v0 problem, so it was decided to test the new learning algorithm on this problem as well. For the CartPole-v0 problem it turned out to be sufficient to turn off weight changes if the network performed $S_{\max}$ actions by changing the coefficient K according to the equation (5):

$$K = \begin{cases} 0, & \text{if} \quad \max(S_i, R_{i+1}) = 200; \\ 1, & \text{if} \quad \max(S_i, R_{i+1}) < 200. \end{cases} \tag{5}$$

Figure 5 shows the history of the reward change and the number of steps played in each episode. Figure 6 shows how the trained network passes the test, in which it must keep the pole in the equilibrium state above 195 steps for 100 episodes. As a result, it is clear that the network successfully solves the problem, spending no more than 10 000 episodes on training.

After successful training to solve the problem for version v0, this algorithm was tested on the CartPole-v1 problem. Unlike version v0, it was not possible to achieve training without decreasing λ depending on the reward, so for successful training it was necessary to change the coefficient K according to Eq. (6):

$$K = \begin{cases} 0, & \text{if} \quad \max(S_i, R_{i+1}) = 500; \\ \frac{100}{\max(S_i, R_{i+1})}, \\ \text{if} \quad \max(S_i, R_{i+1}) < 500. \end{cases} \tag{6}$$

As it can be seen, lowering λ helps network keep current weights and slowly optimize them to keep

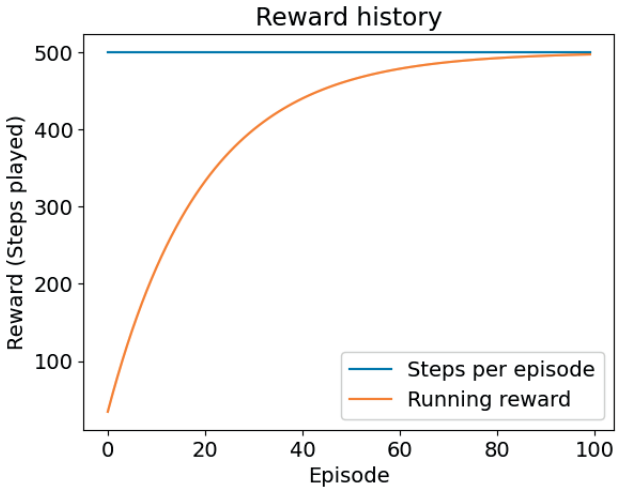

Fig. 8. Running reward of CartPole-v1 test simulation. Orange line shows that at the end of 100 episodes reward is higher than 475, so task is considered as solved.

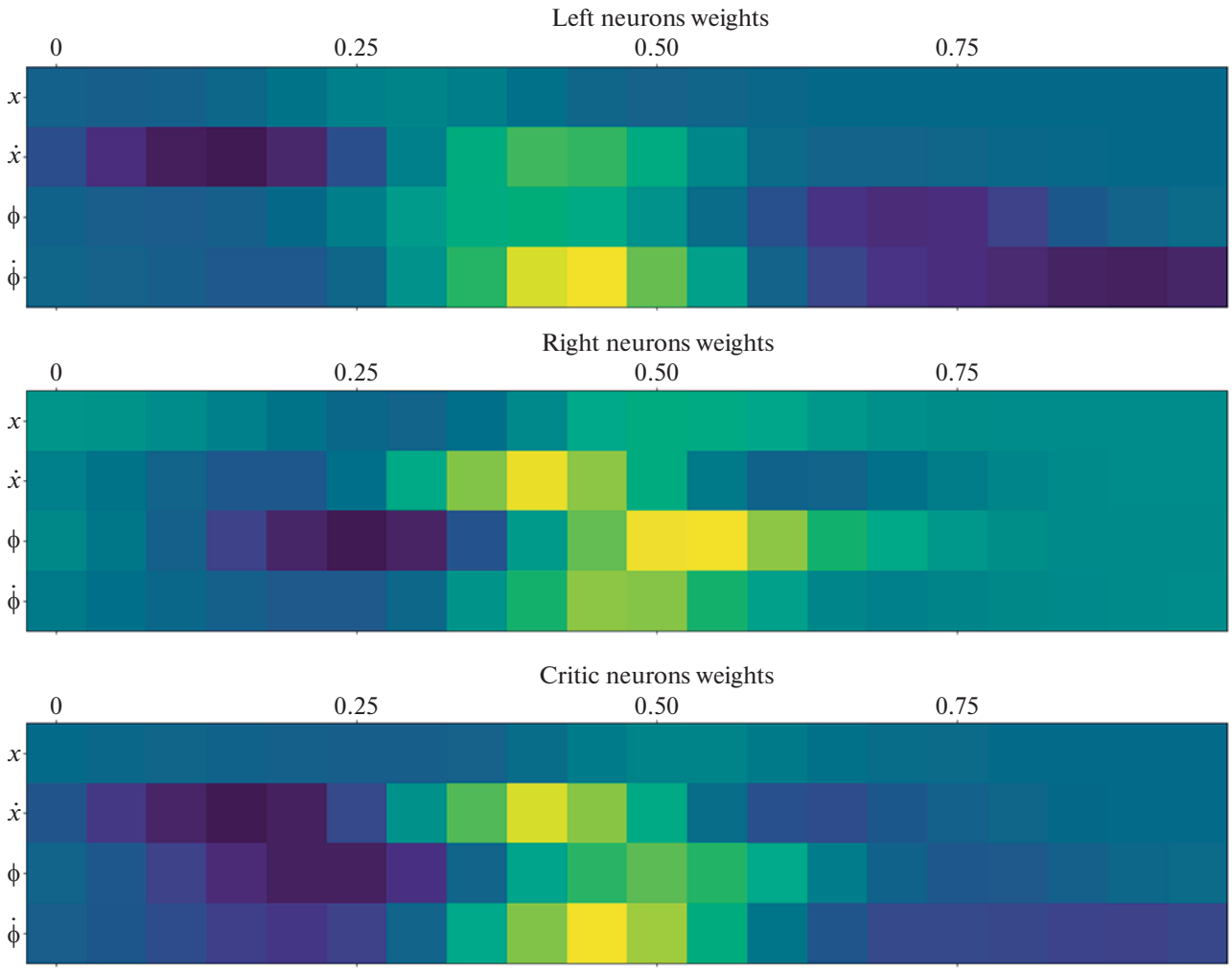

Fig. 9. Weights of the action neurons and critic neurons after successfull learnig of CartPole-v1 task. Each state variable is converted into row of receptive fields, where each field denotes particular value of variable.

pole in equilibrium state longer (Fig. 7) and also allows network to successfully solve task (Fig. 8).

The final weights of the neurons are shown in Fig. 9.

4. CONCLUSIONS

The proposed learning algorithm is based on the modulation of synaptic plasticity of the neurons responsible for actions and the critic neuron depending on the difference in the value of the expected reward value. The results demonstrate the success of this type of learning for both the CartPole-v0 and CartPole-v1 tasks. Solving the CartPole-v1 task requires less time steps due to changing the multiplier of plasticity modulation depending on the running reward which helps the network to establish equilibrium during the required number of steps. The amount of episodes needed to solve Carpole-v1 task is comparable to thus in paper [27] and requires less amount spikes. Also, weights converge twice faster than in [32] having same amount of spikes.

As a development of the study, it is necessary to apply this algorithm to such classical tasks as MountainCar and Acrobot, and also to expand the results by using memristive plasticity instead of classical STDP.

The proposed method is based on temporal encoding, which leads to lower energy consumption compared to frequency encoding and faster decision making. Thus, the SNN model needs 50 ms to predict an action at the inference step and 50 ms to learn from this action. This allows us to talk about the algorithm's operation almost in real time. Further research will allow us to develop autonomous devices that could be controlled and trained directly by neuromorphic chips based on memristors.

Algorithm 1. Reinforcement learning algorithm for actor-critic spiking neural networks

Input: Initial environment state s, neuron parameters, plasticity parameters, initial distribution of weights
Parameter: t_h, h, $R_{\max}$
Output: network weights

Initialize neural network: neurons, synapses and initial weights.
$K \leftarrow 1$
while R < $R_{\max}$ **do**
  for $S \leftarrow 1$ to $S_{\max}$ **do**
    acquire environment state s_t
    convert s_t to vector of spike times t using GRF (Sec. 2.3)
    apply s_t to the network by simulating it for t_h and acquire an action a_t using zero synaptic weight change amplitude
    get critic neuron reward r_t of s_t
    implement the action a_t in the environment and acquire the state s_{t+1}
    get critic neuron reward r_{t+1} of s_{t+1}
    calculate $\Delta r = r_{t+1} - r_t$
    modulating synaptic plasticity depending on Δr, running reward R_{i+1} and games played
    apply the state s_t along with modulating amplitude of synaptic plasticity of the neuron caused the action a_t by simulating network for t_h
    calculate running reward R_{i+1}
    if $\theta > \theta_{\max}$ **or** $x > x_{\max}$ **then**
      calculate coefficient $K(S_i, R_{i+1})$
      calculate amplitude of synaptic plasticity $\lambda = \lambda_0 \Delta r K$
      break
    end if
  end for
end while
return weight distribution, output spike times

ACKNOWLEDGMENTS

This work has been carried out using computing resources of the federal collective usage center Complex for Simulation and Data Processing for Mega-science Facilities at NRC "Kurchatov Institute," http://ckp.nrcki.ru/.

FUNDING

This work was supported by the Russian Science Foundation, project no. 23-11-00260, https://rscf.ru/project/23-11-00260/.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. J. Zhu, T. Zhang, Yu. Yang, and R. Huang, Appl. Phys. Rev. **7**, 11312 (2020). https://doi.org/10.1063/1.5118217
2. D. Ielmini and S. Menzel, in *Resistive Switching*, Ed. by D. Ielmini and R. Waser (Wiley, 2016), p. 317. https://doi.org/10.1002/9783527680870.ch11
3. Yu. V. Pershin and M. Di Ventra, Neural Networks **23**, 881 (2010). https://doi.org/10.1016/j.neunet.2010.05.001
4. K. Berggren, Q. Xia, K. K. Likharev, D. B. Strukov, H. Jiang, T. Mikolajick, D. Querlioz, M. Salinga, J. R. Erickson, Sh. Pi, F. Xiong, P. Lin, C. Li, Yu. Chen, Sh. Xiong, B. D. Hoskins, M. W. Daniels, A. Madhavan, J. A. Liddle, J. J. Mcclelland, Yu. Yang, J. Rupp, S. S. Nonnenmann, K.-T. Cheng, N. Gong, M. A. Lastras-Montaño, A. A. Talin, A. Salleo, B. J. Shastri, T. F. De Lima, P. Prucnal, A. N. Tait, Yi. Shen, H. Meng, Ch. Roques-Carmes, Z. Cheng, H. Bhaskaran, D. Jariwala, H. Wang, J. M. Shainline, K. Segall, J. J. Yang, K. Roy, S. Datta, and A. Raychowdhury, Nanotechnology **32**, 012002 (2020). https://doi.org/10.1088/1361-6528/aba70f
5. B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, IEEE Signal Process. Mag. **36**, 97 (2019). https://doi.org/10.1109/msp.2019.2933719
6. P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, Ch. Guo, Yu. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, Science **345**, 668 (2014). https://doi.org/10.1126/science.1254642
7. M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Yo. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, Sh. Jain, Yu. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. Mccoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Yo. Yang, and H. Wang, IEEE Micro **38**, 82 (2018). https://doi.org/10.1109/mm.2018.112130359
8. J. Pei, L. Deng, S. Song, M. Zhao, Yo. Zhang, Sh. Wu, G. Wang, Zh. Zou, Zh. Wu, W. He, F. Chen, N. Deng, S. Wu, Yu. Wang, Yu. Wu, Zh. Yang, Ch. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Yu. Xie, and L. Shi, Nature **572**, 106 (2019). https://doi.org/10.1038/s41586-019-1424-8

9. P. Dayan and L. Abbott, *Theoretical Neuroscience* (MIT Press, Cambridge, MA, 2001).
10. Zh. Wang, S. Joshi, S. Savel'ev, W. Song, R. Midya, Yu. Li, M. Rao, P. Yan, Sh. Asapu, Ye. Zhuo, H. Jiang, P. Lin, C. Li, J. H. Yoon, N. K. Upadhyay, J. Zhang, M. Hu, J. P. Strachan, M. Barnell, Q. Wu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, Nat. Electron. **1**, 137 (2018). https://doi.org/10.1038/s41928-018-0023-2
11. G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, and D. Ielmini, Sci. Rep. **7**, 5288 (2017). https://doi.org/10.1038/s41598-017-05480-0
12. *Resistive Switching: From Fundamentals of Nanoionic Redox Processes to Memristive Device Applications*, Ed. by D. Ielmini and R. Waser (Wiley, 2015). https://doi.org/10.1002/9783527680870
13. M. Hu, C. E. Graves, C. Li, Yu. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan, Adv. Mater. **30**, 1705914 (2018). https://doi.org/10.1002/adma.201705914
14. Zh. Wang, C. Li, W. Song, M. Rao, D. Belkin, Yu. Li, P. Yan, H. Jiang, P. Lin, M. Hu, J. P. Strachan, N. Ge, M. Barnell, Q. Wu, A. G. Barto, Q. Qiu, R. S. Williams, Q. Xia, and J. J. Yang, Nat. Electron. **2**, 115 (2019). https://doi.org/10.1038/s41928-019-0221-6
15. C. Li, Zh. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Yu. Li, P. Lin, M. Hu, N. Ge, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, Nat. Mach. Intell. **1**, 49 (2019). https://doi.org/10.1038/s42256-018-0001-4
16. J. del Valle, J. G. Ramírez, M. J. Rozenberg, and I. K. Schuller, J. Appl. Phys. **124**, 211101 (2018). https://doi.org/10.1063/1.5047800
17. M. Prezioso, F. Merrikh Bayat, B. Hoskins, K. Likharev, and D. Strukov, Sci. Rep. **6**, 21331 (2016). https://doi.org/10.1038/srep21331
18. Ya. Zhang, X. Wang, and E. G. Friedman, IEEE Trans. Circuits Syst. I: Regular Pap. **65**, 677 (2018). https://doi.org/10.1109/tcsi.2017.2729787
19. Q. Xia and J. J. Yang, Nat. Mater. **18**, 309 (2019). https://doi.org/10.1038/s41563-019-0291-x
20. T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, Front. Neurosci. **7**, 2 (2013). https://doi.org/10.3389/fnins.2013.00002
21. A. A. Minnekhanov, B. S. Shvetsov, A. V. Emelyanov, K. Yu. Chernoglazov, E. V. Kukueva, A. A. Nesmelov, Yu. V. Grishchenko, M. L. Zanaveskin, V. V. Rylkov, and V. A. Demin, J. Phys. D: Appl. Phys. **54**, 484002 (2021). https://doi.org/10.1088/1361-6463/ac203c
22. A. A. Minnekhanov, A. V. Emelyanov, D. A. Lapkin, K. E. Nikiruy, B. S. Shvetsov, A. A. Nesmelov, V. V. Rylkov, V. A. Demin, and V. V. Erokhin, Sci. Rep. **9**, 10800 (2019). https://doi.org/10.1038/s41598-019-47263-9
23. E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, Front. Neurosci. **10**, 482 (2016). https://doi.org/10.3389/fnins.2016.00482
24. D. Querlioz, O. Bichler, P. Dollfus, and Ch. Gamrat, IEEE Trans. Nanotechnol. **12**, 288 (2013). https://doi.org/10.1109/TNANO.2013.2250995
25. V. A. Demin, D. V. Nekhaev, I. A. Surazhevsky, K. E. Nikiruy, A. V. Emelyanov, S. N. Nikolaev, V. V. Rylkov, and M. V. Kovalchuk, Neural Networks **134**, 64 (2021). https://doi.org/10.1016/j.neunet.2020.11.005
26. A. Sboev, D. Vlasov, R. Rybka, Yu. Davydov, A. Serenko, and V. Demin, Mathematics **9**, 3237 (2021). https://doi.org/10.3390/math9243237
27. N. Frémaux, H. Sprekeler, and W. Gerstner, PLoS Comput. Biol. **9**, e1003024 (2013). https://doi.org/10.1371/journal.pcbi.1003024
28. E. Dzhivelikian, A. Latyshev, P. Kuderov, and A. I. Panov, Brain Inf. **9**, 8 (2022). https://doi.org/10.1186/s40708-022-00156-6
29. F. Zhao, Yi. Zeng, G. Wang, J. Bai, and B. Xu, Cognit. Comput. **10**, 296 (2018). https://doi.org/10.1007/s12559-017-9511-3
30. R. S. Sutton, Mach. Learn. **3**, 9 (1988). https://doi.org/10.1007/bf00115009
31. D. Auge, J. Hille, E. Mueller, and A. Knoll, Neural Process. Lett. **53**, 4693 (2021). https://doi.org/10.1007/s11063-021-10562-2
32. D. Vlasov, A. Minnekhanov, R. Rybka, Yu. Davydov, A. Sboev, A. Serenko, A. Ilyasov, and V. Demin, Neural Networks **166**, 512 (2023). https://doi.org/10.1016/j.neunet.2023.07.031
33. N. A. Kudryashov, R. B. Rybka, and A. G. Sboev, Appl. Math. Lett. **76**, 142 (2018). https://doi.org/10.1016/j.aml.2017.08.013
34. A. V. Emelyanov, D. A. Lapkin, V. A. Demin, V. V. Erokhin, S. Battistoni, G. Baldi, A. Dimonte, A. N. Korovin, S. Iannotta, P. K. Kashkarov, and M. V. Kovalchuk, AIP Adv. **6**, 111301 (2016). https://doi.org/10.1063/1.4966257
35. A. Sboev, R. Rybka, A. Serenko, D. Vlasov, N. Kudryashov, and V. Demin, Procedia Computer Science **123**, 432 (2018). https://doi.org/10.1016/j.procs.2018.01.066

36. D. L. Manna, A. Vicente-Sola, P. Kirkland, T. J. Bihl, and G. Di Caterina, Neuromorphic Comput. Eng. **2**, 044009 (2022).
https://doi.org/10.1088/2634-4386/ac999b

37. A. Sboev, A. Serenko, R. Rybka, and D. Vlasov, Math. Methods Appl. Sci. **43**, 7802 (2020).
https://doi.org/10.1002/mma.6241

38. A. Sboev, D. Vlasov, R. Rybka, Yu. Davydov, A. Serenko, and V. Demin, Mathematics **9**, 3237 (2021).
https://doi.org/10.3390/math9243237

39. G.-Q. Bi and M.-M. Poo, Annu. Rev. Neurosci. **24**, 139 (2001).
https://doi.org/10.1146/annurev.neuro.24.1.139

40. R. Gütig and H. Sompolinsky, Nat. Neurosci. **9**, 420 (2006).
https://doi.org/10.1038/nn1643

41. X. Wang, Z.-G. Hou, F. Lv, M. Tan, and Yo. Wang, Neurocomputing **134**, 230 (2014).
https://doi.org/10.1016/j.neucom.2013.07.055

42. Q. Yu, H. Tang, K. Ch. Tan, and H. Yu, Neurocomputing **138**, 3 (2014).
https://doi.org/10.1016/j.neucom.2013.06.052

43. A. G. Barto, R. S. Sutton, and Ch. W. Anderson, IEEE Trans. Syst., Man, Cybern. **SMC–13**, 834 (1983).
https://doi.org/10.1109/tsmc.1983.6313077