mathematics

MDPI

Article

Modeling the Dynamics of Spiking Networks with Memristor-Based STDP to Solve Classification Tasks

Alexander Sboev [1,2,*], Danila Vlasov [1], Roman Rybka [1], Yury Davydov [1], Alexey Serenko [1] and Vyacheslav Demin [1]

[1] National Research Centre "Kurchatov Institute", 123182 Moscow, Russia; vfked0d@gmail.com (D.V.); rybkarb@gmail.com (R.R.); davydov.workbox@gmail.com (Y.D.); serenko@phystech.edu (A.S.); Demin_VA@nrcki.ru (V.D.)

[2] Moscow Engineering Physics Institute, National Research Nuclear University, 115409 Moscow, Russia

* Correspondence: Sboev_AG@nrcki.ru

Abstract: The problem with training spiking neural networks (SNNs) is relevant due to the ultra-low power consumption these networks could exhibit when implemented in neuromorphic hardware. The ongoing progress in the fabrication of memristors, a prospective basis for analogue synapses, gives relevance to studying the possibility of SNN learning on the base of synaptic plasticity models, obtained by fitting the experimental measurements of the memristor conductance change. The dynamics of memristor conductances is (necessarily) nonlinear, because conductance changes depend on the spike timings, which neurons emit in an all-or-none fashion. The ability to solve classification tasks was previously shown for spiking network models based on the bio-inspired local learning mechanism of spike-timing-dependent plasticity (STDP), as well as with the plasticity that models the conductance change of nanocomposite (NC) memristors. Input data were presented to the network encoded into the intensities of Poisson input spike sequences. This work considers another approach for encoding input data into input spike sequences presented to the network: temporal encoding, in which an input vector is transformed into relative timing of individual input spikes. Since temporal encoding uses fewer input spikes, the processing of each input vector by the network can be faster and more energy-efficient. The aim of the current work is to show the applicability of temporal encoding to training spiking networks with three synaptic plasticity models: STDP, NC memristor approximation, and PPX memristor approximation. We assess the accuracy of the proposed approach on several benchmark classification tasks: Fisher's Iris, Wisconsin breast cancer, and the pole balancing task (CartPole). The accuracies achieved by SNN with memristor plasticity and conventional STDP are comparable and are on par with classic machine learning approaches.

Keywords: spiking neural networks; synaptic plasticity; spike-timing-dependent plasticity; memristor

Citation: Sboev, A.; Vlasov, D.; Rybka, R.; Davydov, Y.; Serenko, A.; Demin, V. Modeling the Dynamics of Spiking Networks with Memristor-Based STDP to Solve Classification Tasks. *Mathematics* **2021**, *9*, 3237. https://doi.org/10.3390/math9243237

Academic Editors: Nikolai A. Kudryashov and Cornelio Yáñez Márquez

Received: 6 November 2021
Accepted: 10 December 2021
Published: 14 December 2021

1. Introduction

A variety of problems surround the phenomena or dynamical processes that cannot be described by explicit laws expressed in differential equations. Such tasks could be solved with the help of data-driven modeling, which forms an implicit model of the process of interest by learning from the observed data. An especially relevant direction in data-driven modeling involves spiking neural networks (SNNs) [1–3], an inherent characteristic of which is the nonlinearity in the temporal dynamics of neurons receiving and transmitting spikes and the dynamics of the synaptic weights during learning. The dynamics of spiking neurons is described by nonlinear differential equations: the membrane potential of a neuron receives non-differentiable pulses when input spikes arrive and is instantaneously reset to its resting value upon emitting an output spike.

The practical relevance of SNNs involves the ultra-low power consumption these networks could exhibit when implemented in neuromorphic hardware [4,5]. For instance, the digital neuromorphic chip TrueNorth [6] spends only 26 pJ for transmitting an impulse

(spike) from neuron-to-neuron. Devices in which synapses (and possibly neurons too) are implemented in an analogue fashion can be even more efficient [7]. The prospective element base for the analogue implementation of a synapsis is a memristor [8,9].

This gives relevance toward developing spiking neural network models with learning based on synaptic plasticity mechanisms that model the conductance change of a memristor. A number of memristor plasticity models have been obtained so far, backed by experimental measurements, in which the drift of the conductance of a memristor depends nonlinearly on its current conductance and on the time difference between presynaptic and postsynaptic spikes [10–14]. Spiking networks with the plasticity approximating nanocomposite (NC) memristors $(CoFeB)_x(LiNbO_3)_{1-x}$ were shown to classify the MNIST handwritten digits [15]. Recently, a highly-plastic poly-p-xylylene (PPX) memristor was created [16], which makes it relevant to study the possibility of learning about SNNs, with plasticity modeling that type of memristor.

This paper considers three synaptic plasticity models: the model of the PPX memristor plasticity obtained by approximation of its experimental measurements, the existing NC memristor plasticity model [15], and the additive spike-timing-dependent plasticity, which was shown to resemble the plasticity of various types of memristors [17,18].

The aim of this paper is to numerically solve the learning dynamics of the spiking neural network model with the aforementioned plasticity mechanisms, to obtain weights established after learning, and to obtain the times of output spikes for given input spikes, which are then decoded into classes to solve a classification task.

Unlike existing works devoted to SNN learning with memristor plasticity models [15,17,19–21], which are based on frequency encoding of the input data, we use temporal encoding, in which the information is contained in the timings of input spike patterns, as it requires fewer spikes and, thus, less energy.

For the NC and PPX memristor plasticity models (described in Section 2.2), we show in Section 3.1 that a neuron memorizes repetitive spike patterns. Based on this, an algorithm for training a spiking neural network with temporal encoding is proposed in Section 2.5. The performance of the algorithm is tested in Section 3.2 on benchmark classification problems.

2. Materials and Methods

2.1. Neuron Model

Keeping in mind the prospective possibility of hardware implementation, we strive for a simple neuron model. We thus use the leaky integrate-and-fire model [22] for the neuron dynamics, in which the neuron has one state variable, the membrane potential $V(t)$, which obeys the following dynamics as soon as it is below the threshold V_{th}:

$$\frac{dV}{dt} = -\frac{V(t) - V_{\text{rest}}}{\tau_{\text{m}}} + \frac{I_{\text{syn}}(t) + I_{\text{ext}}(t)}{C_{\text{m}}}. \tag{1}$$

The neuron is considered to fire an output spike when $V(t)$ exceeds V_{th}, after which V is instantaneously reset to 0, and during the refractory period t_{ref} the neuron is unable to fire spikes.

$I_{\text{ext}}(t)$ is the external stimulation current applied during training, described in Section 2.5. $I_{\text{syn}}(t)$ is the incoming postsynaptic current, summed over currents $I_{\text{syn},i}(t)$ coming from the neuron's input synapses:

$$I_{\text{syn}}(t) = \sum_i I_{\text{syn},i}(t), \qquad \frac{dI_{\text{syn},i}}{dt} = -\frac{I_{\text{syn},i}(t)}{\tau_{\text{syn}}} + w_i(t)\frac{q_{\text{syn}}}{\tau_{\text{syn}}} S_{\text{pre},i}(t - t_{\text{delay}}). \tag{2}$$

Here, $S_{\text{pre},i}(t)$ is equal to 1 when a presynaptic spike arrives at the i-th input synapse of the neuron, and to 0 otherwise. The arrivals of presynaptic spikes are governed by the input encoding algorithm described in Section 2.4. t_{delay} is the delay for transmitting a presynaptic spike to the postsynaptic neuron, in our simulations equal to the integration timestep dt = 0.1 ms. C_{m} = 1 pF, q_{syn} = 5 fC, τ_{syn} = 5 ms. The constants V_{th}, τ_{m}, and t_{ref} are adjusted for each particular classification task and presented in Section 3.

The dimensionless synaptic weight $0 \leqslant w_i(t) \leqslant 1$ changes after each presynaptic and postsynaptic spike in accordance with the plasticity model, as defined in Section 2.2.

2.2. Plasticity Models

2.2.1. Additive Spike-Timing-Dependent Plasticity

For the sake of comparison, in addition to memristive plasticity models, which will be presented in the next sections, we perform numerical experiments with the conventional STDP [23] in its additive form, where the synaptic weight change Δw does not depend on the current weight w, and only depends on the time interval Δt from the arrival of a presynaptic spike to emitting the postsynaptic spike:

$$\Delta w = \begin{cases} -A^- \cdot \exp\left(\frac{\Delta t}{\tau^-}\right) & \text{if } \Delta t < 0; \\ A^+ \cdot \exp\left(-\frac{\Delta t}{\tau^+}\right) & \text{if } \Delta t > 0. \end{cases} \tag{3}$$

Here, following the existing literature [24], $\tau^+ = 20$ ms, $\tau^- = 20$ ms, $A^+ = A^- = 0.01$.

Solving the synapse dynamics is performed with the help of two more state variables for each synapse i, its presynaptic and postsynaptic eligibility traces [25] x_i and y_i:

$$\begin{cases} \frac{dx_i}{dt} = -\frac{x_i(t)}{\tau_+} + S_{\text{pre},i}(t), \\ \frac{dy_i}{dt} = -\frac{y_i(t)}{\tau_-} + S_{\text{post}}(t). \\ \frac{dw_i}{dt} = \max\left(A^- \cdot y_i(t) \cdot S_{\text{pre},i}(t),\ 1 - w_i\right) + \\ \qquad + \min\left(A^+ \cdot x_i(t) \cdot S_{\text{post}}(t),\ w\right). \end{cases} \tag{4}$$

2.2.2. Nanocomposite Memristor Plasticity

The plasticity model for nanocomposite memristors $(CoFeB)_x(LiNbO_3)_{1-x}$ is borrowed from the literature [15]:

$$\Delta w(\Delta t) = \begin{cases} A^+ \cdot w \cdot \left[1 + \tanh\left(-\frac{\Delta t - \mu_+}{\tau_+}\right)\right] & \text{if } \Delta t > 0; \\ A^- \cdot w \cdot \left[1 + \tanh\left(\frac{\Delta t - \mu_-}{\tau_-}\right)\right] & \text{if } \Delta t < 0. \end{cases} \tag{5}$$

The constants are kept as in the original literature [15]: $A^+ = 0.074$, $A^- = -0.047$, $\mu^+ = 26.7$ ms, $\mu^- = -22.3$ ms, $\tau^+ = 9.3$ ms, $\tau^- = 10.8$ ms.

The spike timing dependence curves for different conductance values are depicted in Figure 1A.

2.2.3. Model of Poly-p-Xylylene Memristors

PPX-based memristors, in contrast to NC-based memristors, demonstrate resistive switching driven by electrochemical metallization mechanism: conductive filaments are formed in them due to electromigration of metal ions [16]. This leads to a slightly different shape of the spike timing dependence curves.

We fitted the experimental dependences of the change in synaptic conductance on the time interval Δt between presynaptic and postsynaptic splices for PPC memristors using the following function:

$$\Delta w(\Delta t) = \begin{cases} \frac{|\Delta t|}{\tau} \alpha^+ e^{-\beta^+ \left(\frac{w_{\max} - w}{w_{\max} - w_{\min}}\right)} e^{-\gamma^+ \left(\frac{\Delta t}{\tau}\right)^2} & \text{if } \Delta t > 0; \\ \frac{|\Delta t|}{\tau} \alpha^- e^{-\beta^- \left(\frac{w - w_{\min}}{w_{\max} - w_{\min}}\right)} e^{-\gamma^- \left(\frac{\Delta t}{\tau}\right)^2} & \text{if } \Delta t < 0. \end{cases} \tag{6}$$

Here $\tau = 10$ ms, $\alpha^+ = 0.32$, $\alpha^- = 0.01$, $\beta^+ = 2.21$, $\beta^- = -5.97$, $\gamma^+ = 0.03$, $\gamma^- = 0.15$, $w_{\max} = 1$, $w_{\min} = 0$.

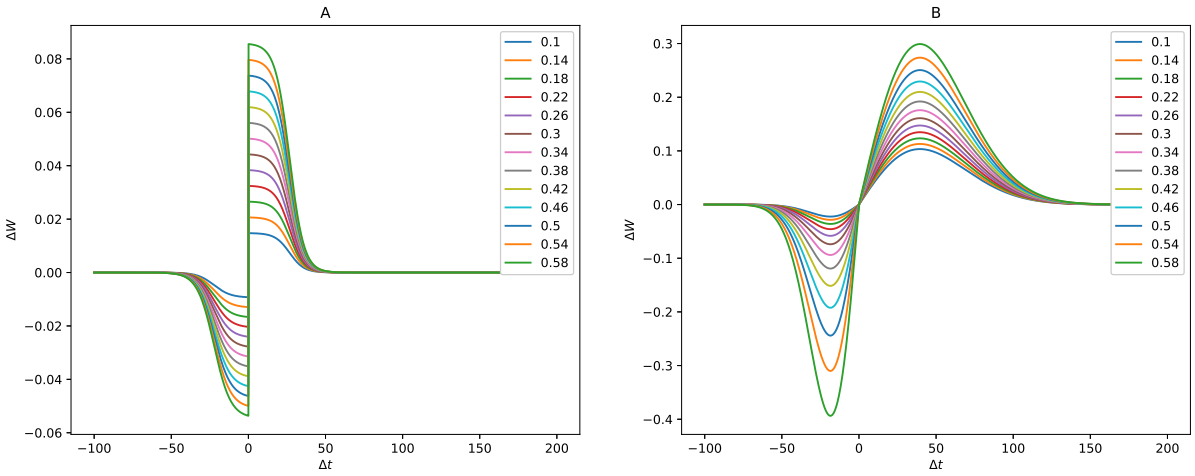

Figure 1. Spike timing dependence curves: the dependence of the change Δw in synaptic conductance on the interval Δt between a presynaptic spike and a postsynaptic spike for different current synaptic conductance values w. (**A**): for the nanocomposite memristors, redrawn from the original paper [15], (**B**): for poly-p-xylylene memristors.

The weight-dependent exponents in Equation (6) express the experimentally observed dependence of the change in synaptic conductance on the initial conductance value. Similar dependencies have already been applied in some works on memristic conductivity, in particular in [26]. Parameters $\alpha^+, \alpha^-, \beta^+, \beta^-, \gamma^+, \gamma^-$ were determined from the experimentally obtained dependencies in three stages: at the first stage, experimental dependencies were approximated by cubic splines. At the second stage, the obtained spline curves were approximated by the function above (see Equation (6)) for each set of experimental data by the nonlinear least squares method At the third stage, the best set of parameters was chosen based on the maximum possible values of R^2. The experimental data consisted of measurements of the dependence of the change in synaptic conductance on Δt for four different initial conductance values, for each of which, measurements were performed five times, after which the results were averaged. The results of the experiments and approximations are shown in Figure 1B.

2.3. Network Model Implementation

Overall, the network is defined by the following system of equations:

$$\begin{cases} \text{For each neuron } j\text{:} \\ \quad V_j(t) = \int\limits_{\hat{t}_j}^{t} \exp\left(-\dfrac{t-t'}{\tau_\text{m}}\right) \cdot \Big(I_{\text{ext, for neuron } j}(t') + I_{\text{syn, for neuron } j}(t')\Big)dt', \\ \quad S_{\text{post,j}}(t) = \Theta\big(V_j(t) - V_\text{th}\big) \cdot \Theta\big(t - \hat{t}_j - t_\text{ref}\big); \\ \text{For each input component } i\text{:} \\ \quad S_{\text{pre},i}(t) = \sum\limits_{t^i_\text{input}} \delta\Big(t - t^i_\text{input}\Big); \\ \text{For each input synapse } i \text{ of each neuron } j\text{:} \\ \quad \dfrac{dw_{ij}}{dt} = \text{Plasticity}\big(w_{ij}, S_{\text{pre},i}, S_{\text{post,j}}\big). \end{cases} \tag{7}$$

Here, the formal solution for a neuron's potential $V_j(t)$ is presented [27,28], starting from the moment $\hat{t}_j$ of its most recent spike. The initial conditions are $V_j(\hat{t}_j) = 0$, $w_{ij}(0) = w_\text{init}$. The times t^i_input of the presynaptic spikes arriving from each input i during presenting every input vector are defined in Section 2.4. I_syn are defined in Equation (2). Plasticity refers to one of the models (3), (5), or (6). Θ is the heaviside step function.

Solving the network dynamics is performed numerically in a piecewise manner: $V_j(t)$ is obtained over an interval during which $S_{\text{post,j}}(t)$ and all $S_{\text{pre},i}(t)$ equal 0. When a postsynaptic spike occurs, w_{ij} is updated in accordance with the plasticity model, and $\hat{t}_j$ is updated to equal the current value of t. When a presynaptic spike arrives, w_{ij} is updated, and the integration continues.

Simulations are carried out with the help of the NEural Simulation Tool (NEST) library [29].

2.4. Input Preprocessing and Encoding

Before presenting input data to the SNN, it is normalized by applying L2 norm or MinMaxScale (https://scikit-learn.org/stable/modules/preprocessing.html, accessed on 13 October 2021) depending on the dataset (see Section 2.6), and then processed by Gaussian receptive fields [30–32]. The latter converts an input vector $\vec{x}$ of dimension N, a vector of dimension $N \cdot M$, where M is the number of receptive fields. Each component x^i is transformed into M components $g(x^i, \mu_0), \ldots, g(x^i, \mu_M)$, where $g(x^i, \mu_j) = \exp\left(\frac{(x^i - \mu_j)^2}{\sigma^2}\right)$. Here, $\mu_j = X^i_{\min} + (X^i_{\max} - X^i_{\min}) \cdot \frac{j}{M-1}$ is the center of the j-th receptive field, $X^i_{\max}$ and $X^i_{\min}$ are the maximal and minimal values of the i-th component among all vectors of the training set, which are 1 and 0, respectively, if MinMaxScale normalization is applied. M is chosen to be 20 in all experiments.

After preprocessing, the vector obtained is encoded into a pattern of spikes to present to the input synapses of the network. Each component x^i of the preprocessed vector is represented by one spike arriving at he i-th input synapse at time $t^i_{\text{input}} = t_h(1 - x^i)$, relative to the beginning of presenting that input vector, where t_h is the duration of presenting one vector. That way, the particularities of a class of input vectors are characterized by a few of the earliest input spikes, which, in turn, correspond to the receptive fields typical to that class.

2.5. Learning Algorithm

To solve multi-class classification tasks, on the base of local plasticity tasks, the learning algorithm should be designed so that each neuron learns specifically the class it is assigned to. To achieve that, we use a learning algorithm in which neurons memorize their classes induced by a reinforcing signal (see Algorithm 1).

The network consists of as many neurons as there are classes in the classification problem; the neurons are connected with each other by non-plastic inhibitory synapses with fixed weights w_{inh} (see Figure 2).

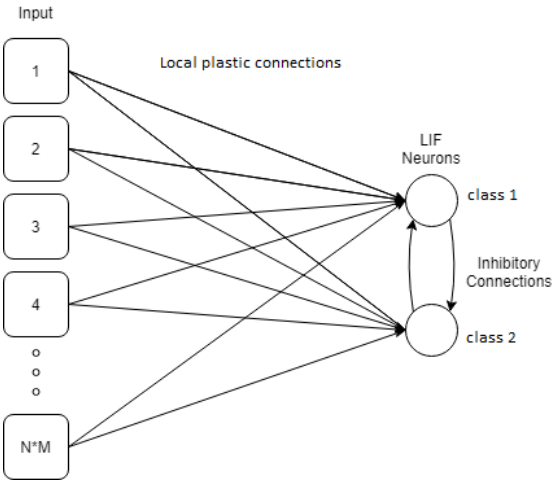

Figure 2. The spiking neural network topology.

At the training stage, the neurons receive spike patterns encoding vectors of classes of the training sample. The neuron that corresponds to the class of the input sample being fed at the moment is stimulated by setting a high positive I_{ext} for a short period, starting from $x_i^{\min} + t_{\text{shift}}$, where $x_i^{\min}$ is the beginning of presenting an input vector. The value of I_{ext} is chosen such that it causes an immediate output spike. The spike induced by such a stimulation will lead to amplification, according to the rule of local plasticity, of those inputs that receive spikes at earlier moments of time. To decrease the probability of excitation of other neurons and prevent their synaptic weights from growing while giving examples of classes that are not assigned to them at the learning stage, the threshold is set so that only the trained neuron spikes in response to reinforcing signal. The class of the example is determined by the neuron that generated the spike earliest.

Algorithm 1 Learning algorithm

Input: matrix of preprocessed input objects X, vector of object classes Y, neuron parameters, plasticity parameters, initial distribution of weights
Parameter: N_epochs, t_h, h
Output: network weights

1: Initialize neural network: neurons, synapses and initial weights.
2: Define input spike patterns with the duration t_h.
3: **for** each x_i in X **do**
4: search for a minimal value of $x_i^{\min}$.
5: define the time since the beginning of the reinforcing signal as $x_i^{\min} + t_{\text{shift}}$, where t_{shift} is a reinforcing signal temporal shift.
6: define the termination time of the reinforcing signal as $x_i^{\min}$ + t_{shift} + 2 * dt, where dt is the simulation timestep.
7: Set an amplitude for the reinforcing signal.
8: **end for**
9: **for** k in N_epochs **do**
10: Set input spikes at the generators.
11: Set teacher current impulse times at the generators.
12: Simulate a training epoch.
13: For the next sample, times of input spikes and teacher current impulse times are shifted on a time period equal to the epoch simulation time.
14: **end for**
15: **return** weight distribution, output spike times.

2.6. Datasets

Two benchmark classification problems are considered: Fisher's Iris and Wisconsin breast cancer.

The dataset of Fisher's Iris consists of 150 flowers, described by four traits: the length and width of the sepal and petals in centimeters. The specimens belong to three different classes of 50 specimens each, corresponding to three species: *Iris setosa*, *Iris virginica*, and *Iris versicolor*. The first class is linearly separable from the second and third, while the second and third are not linearly separable.

The breast cancer dataset collected at the University of Wisconsin consists of 569 samples, 357 of which are classified as "benign" and 212 as "malignant". Each sample in the dataset represents cell characteristics from a digitized image of a fine needle aspiration breast biopsy. The input vector of length 30 is composed of the mean value (among all cells), the standard deviation, and the extreme values of each of the 10 cell nucleus characteristics—radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.

Pole balancing [33] is originally a reinforcement learning task. However, creating a reinforcement learning algorithm for SNNs with memristive plasticity will be included in future work. As a preliminary step for that, we here consider it as a classification task.

In this task, the objective is to hold a massive pole attached to a moving cart by a hinge for a given number of episodes (at least 195 out of 200) by changing the position of the carriage. The environment is characterized by four parameters: coordinate and speed of the carriage, as well as angle of deviation from the vertical and angular velocity of the pole (x, $\dot{x}$, ϕ and $\dot{\phi}$). The control action which the network should predict applies a force of 1 N to the carriage in the left or right direction.

To convert this task into a classification problem, we collected a reference set of environmental states and control actions with the help of an artificial neural network, with one hidden layer of two neurons to the task. This network was trained using the RL algorithm Policy Gradient (https://github.com/norse/norse/blob/master/norse/task/cartpole.py, accessed on 24 October 2021) until the average number of episodes (carriage movements), during which the pole remained in an acceptable position, was equal to 198 (out of 200 episodes). After the artificial neural network was successfully trained, it was run in the CartPole environment without training, and the decisions it made at each step and their corresponding environment states were recorded. A total of 100 runs were performed, which resulted in the collection of 1949 input-output pairs. The collected set of pairs was used to train the spiking neural network.

3. Results

3.1. Memorizing Repeating Patterns

The first experiment was aimed at testing the underlying effect necessary for learning with temporal encoding. This effect was shown previously [34] for STDP: if a neuron gets a repeating spike pattern among Poisson noise, the neuron will gradually become selectively sensitive to this pattern. The times of spikes emitted by the neuron in response to the pattern will gradually become closer to the beginning of its presentation.

We tested this effect by feeding a single neuron with a single vector from the Fisher's Iris dataset, interspersed with random Poisson spike sequences. When a repeating spike pattern is presented to a neuron, the synaptic weights change, so that the neuron generates spike earlier, related to the start of pattern presentation (Figure 3). The spike time eventually established depends on the value of the neuron threshold. At the same time, the neuron gradually stops spiking during presenting Poisson noise. Plasticity modeling PPX memristors is less robust to noise due to the high value of its time window constant τ.

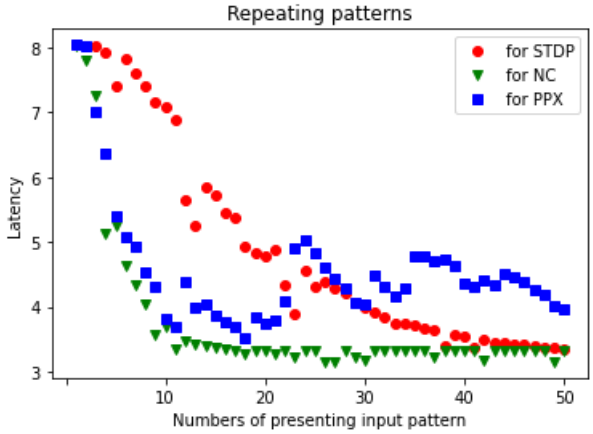

Figure 3. Reduction over time of the delay between the start of the repetitive input spike pattern and the output spike of the neuron.

In the next section, the learning algorithm based on the pattern memorization effect confirmed here for all three plasticity models is tested on benchmark classification datasets: Fisher's Iris, Wisconsin breast cancer, and CartPole.

3.2. Classification with SNN

For each dataset, the learning algorithm was applied three times: with STDP, with NC plasticity, and with PPX plasticity. The plasticity model constants were kept unchanged as originally defined. The neuron model and input encoding constants were adjusted when necessary. As a result, the neuron membrane time constant τ_m was found to be 13 ms. The neuron refractoriness period is $t_{ref} = 300$ ms, so that after emitting a spike, it cannot spike again up until the end of the inter-pattern interval $t_h = 400$ ms. The initial weight of excitatory synapses is $w_{init} = 0.5$. The inhibitory weight $w_{inh} = -4$.

The parameters adjusted separately for each task are shown in Table 1 along with the accuracies of solving respective classification tasks. Accuracy is measured by the F1-macro score, since the classes are almost equal by the numbers of input vectors. Mean, minimum, and maximum values are presented over the splits of five-fold cross-validation.

Table 1. Spiking network parameters and F1-score for different classification tasks.

Task	Plasticity	V_{th}, mV	σ	t_{shift}, ms	F1, %		
					mean	min	max
Fisher's Iris	STDP	5	0.005	0	97	93	100
Fisher's Iris	NC	5	0.005	0	97	93	100
Fisher's Iris	PPX	3	0.005	0	97	93	100
Breast cancer	STDP	8	0.005	3.2	94	89	97
Breast cancer	NC	8	0.005	3.2	93	88	96
Breast cancer	PPX	6	0.005	3.2	93	89	96
CartPole	STDP	5	0.01	1.2	66 (199/200)	65	68
CartPole	NC	6	0.01	1.2	63 (199/200)	62	65
CartPole	PPX	5	0.01	1	60 (197/200)	60	68

4. Discussion

The fact that the results were obtained with similar neuron and synapse model parameters indicates a possible applicability of the proposed learning algorithm to other problems, while the parameters reported here could form the initial working range. Still, selecting the network and encoding parameters individually can achieve greater accuracy. For example, for the Wisconsin breast cancer and CartPole tasks, the timing of the reinforcing signal had to be shifted in the positive direction.

The simplicity of the neuron model considered contributes towards the prospective possibility of hardware implementation of the proposed learning algorithm. However, other nonlinear forms of the neuron's response function could be studied in further work.

5. Conclusions

This paper demonstrates the possibility of solving classification tasks using spiking neural network models with synaptic plasticity models that approximate the plasticity of nanocomposite and poly-p-xylylene memristors. The proposed learning algorithm was tested on several benchmark classification tasks: Fisher's Iris, Wisconsin breast cancer, and the pole balancing task. The network hyperparameters were similar for all tasks, which shows the robustness of the approach.

In the future, we plan to test the proposed algorithm on more benchmarks, and analyze more variants of memristive plasticity models.

Author Contributions: Conceptualization, A.S. (Alexander Sboev) and V.D.; funding acquisition, R.R.; investigation, D.V. and Y.D.; methodology, D.V., R.R., and A.S. (Alexey Serenko); project administration, A.S. (Alexander Sboev) and R.R.; resources, V.D.; software, D.V. and Y.D.; supervision, V.D.; writing—original draft, D.V., Y.D., and A.S. (Alexey Serenko); writing—review & editing, A.S. (Alexander Sboev), R.R. and A.S. (Alexey Serenko). All authors have read and agreed to the published version of the manuscript.

Funding: Developing the network model and the learning algorithm and running numerical simulations were supported by Russian Science Foundation grant no. 21-11-00328. Obtaining experimental measurements of memristor conductivity changes in which the memristor plasticity models are based, in Sections 2.2.2 and 2.2.3, was supported by Russian Science Foundation grant no. 20-79-10185.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was 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/, accessed on 13 October 2021.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Paugam-Moisy, H.; Bohte, S.M. Computing with spiking neuron networks. In *Handbook of Natural Computing*; Rozenberg, G., Back, T., Kok, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 335–376. [CrossRef]
2. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Netw.* **2019**, *111*, 47–63. [CrossRef] [PubMed]
3. Taherkhani, A.; Belatreche, A.; Li, Y.; Cosma, G.; Maguire, L.P.; McGinnity, T. A review of learning in biologically plausible spiking neural networks. *Neural Netw.* **2020**, *122*, 253–272. [CrossRef] [PubMed]
4. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **2018**, *38*, 82–99. [CrossRef]
5. Indiveri, G.; Corradi, F.; Qiao, N. Neuromorphic architectures for spiking deep neural networks. In Proceedings of the 2015 IEEE International Electron Devices Meeting, Washington, DC, USA, 7–9 December 2015; pp. 4.2.1–4.2.4.
6. Merolla, P.A.; Arthur, J.V.; Alvarez-Icaza, R.; Cassidy, A.S.; Sawada, J.; Akopyan, F.; Jackson, B.L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **2014**, *345*, 668–673. [CrossRef]
7. Rajendran, B.; Sebastian, A.; Schmuker, M.; Srinivasa, N.; Eleftheriou, E. Low-Power Neuromorphic Hardware for Signal Processing Applications: A review of architectural and system-level design approaches. *IEEE Signal Process. Mag.* **2019**, *36*, 97–110. [CrossRef]
8. Camuñas-Mesa, L.A.; Linares-Barranco, B.; Serrano-Gotarredona, T. Neuromorphic spiking neural networks and their memristor-CMOS hardware implementations. *Materials* **2019**, *12*, 2745. [CrossRef]
9. Saïghi, S.; Mayr, C.G.; Serrano-Gotarredona, T.; Schmidt, H.; Lecerf, G.; Tomas, J.; Grollier, J.; Boyn, S.; Vincent, A.F.; Querlioz, D.; et al. Plasticity in memristive devices for spiking neural networks. *Front. Neurosci.* **2015**, *9*, 51. [CrossRef]
10. Ismail, M.; Chand, U.; Mahata, C.; Nebhen, J.; Kim, S. Demonstration of synaptic and resistive switching characteristics in $W/TiO_2/HfO_2/TaN$ memristor crossbar array for bioinspired neuromorphic computing. *J. Mater. Sci. Technol.* **2022**, *96*, 94–102. [CrossRef]
11. Ryu, J.H.; Mahata, C.; Kim, S. Long-term and short-term plasticity of Ta_2O_5/HfO_2 memristor for hardware neuromorphic application. *J. Alloys Compd.* **2021**, *850*, 156675. [CrossRef]
12. Sboev, A.G.; Emelyanov, A.V.; Nikiruy, K.E.; Serenko, A.V.; Sitnikov, A.V.; Presnyakov, M.Y.; Rybka, R.B.; Rylkov, V.V.; Kashkarov, P.K.; Kovalchuk, M.V.; et al. Self-adaptive STDP-based learning of a spiking neuron with nanocomposite memristive weights. *Nanotechnology* **2019**, *31*, 045201. [CrossRef]
13. Prudnikov, N.V.; Lapkin, D.A.; Emelyanov, A.V.; Minnekhanov, A.A.; Malakhova, Y.N.; Chvalun, S.N.; Demin, V.A.; Erokhin, V.V. Associative STDP-like learning of neuromorphic circuits based on polyaniline memristive microdevices. *J. Phys. D Appl. Phys.* **2020**, *53*, 414001. [CrossRef]
14. Lapkin, D.A.; Emelyanov, A.V.; Demin, V.A.; Berzina, T.S.; Erokhin, V.V. Spike-timing-dependent plasticity of polyaniline-based memristive element. *Microelectron. Eng.* **2018**, *185–186*, 43–47. [CrossRef]
15. Demin, V.; Nekhaev, D.; Surazhevsky, I.; Nikiruy, K.; Emelyanov, A.; Nikolaev, S.; Rylkov, V.; Kovalchuk, M. Necessary conditions for STDP-based pattern recognition learning in a memristive spiking neural network. *Neural Netw.* **2021**, *134*, 64–75. [CrossRef]

16. Minnekhanov, A.A.; Shvetsov, B.S.; Martyshov, M.M.; Nikiruy, K.E.; Kukueva, E.V.; Presnyakov, M.Y.; Forsh, P.A.; Rylkov, V.V.; Erokhin, V.V.; Demin, V.A.; et al. On the resistive switching mechanism of parylene-based memristive devices. *Org. Electron.* **2019**, *74*, 89–95. [CrossRef]
17. Serrano-Gotarredona, T.; Masquelier, T.; Prodromakis, T.; Indiveri, G.; Linares-Barranco, B. STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Front. Neurosci.* **2013**, *7*, 2. [CrossRef]
18. Du, N.; Kiani, M.; Mayr, C.; You, T.; Bürger, D.; Skorupa, I.; Schmidt, O.; Schmidt, H. Single pairing spike-timing dependent plasticity in $BiFeO_3$ memristors with a time window of 25 ms to 125 µs. *Front. Neurosci.* **2015**, *9*, 227. [CrossRef]
19. Qu, L.; Zhao, Z.; Wang, L.; Wang, Y. Efficient and hardware-friendly methods to implement competitive learning for spiking neural networks. *Neural Comput. Appl.* **2020**, *32*, 13479–13490. [CrossRef]
20. Wang, Z.; Joshi, S.; Savel'ev, S.; Song, W.; Midya, R.; Li, Y.; Rao, M.; Yan, P.; Asapu, S.; Zhuo, Y.; et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nat. Electron.* **2018**, *1*, 137–145. [CrossRef]
21. Pedretti, G.; Milo, V.; Ambrogio, S.; Carboni, R.; Bianchi, S.; Calderoni, A.; Ramaswamy, N.; Spinelli, A.; Ielmini, D. Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Sci. Rep.* **2017**, *7*, 1–10. [CrossRef]
22. Burkitt, A.N. A review of the integrate-and-fire neuron model: II. Inhomogeneous synaptic input and network properties. *Biol. Cybern.* **2006**, *95*, 97–112. [CrossRef]
23. Bi, G.Q.; Poo, M.M. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci.* **2001**, *24*, 139–166. [CrossRef]
24. Diehl, P.U.; Cook, M. Unsupervised learning of digit recognition using Spike-Timing-Dependent Plasticity. *Front. Comput. Neurosci.* **2015**, *9*, 99. [CrossRef]
25. Morrison, A.; Diesmann, M.; Gerstner, W. Phenomenological models of synaptic plasticity based on spike timing. *Biol. Cybern.* **2008**, *98*, 459–478. [CrossRef]
26. Querlioz, D.; Dollfus, P.; Bichler, O.; Gamrat, C. Learning with memristive devices: How should we model their behavior? In Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures, San Diego, CA, USA, 8–9 June 2011; pp. 150–156. [CrossRef]
27. Gerstner, W. A framework for spiking neuron models: The spike response model. *Handb. Biol. Phys.* **2001**, *4*, 469–516.
28. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002.
29. Kunkel, S.; Morrison, A.; Weidel, P.; Eppler, J.M.; Sinha, A.; Schenck, W.; Schmidt, M.; Vennemo, S.B.; Jordan, J.; Peyser, A.; et al. NEST 2.12.0. 2017. Available online: https://doi.org/10.5281/zenodo.259534 (accessed on 13 October 2021).
30. Gütig, R.; Sompolinsky, H. The tempotron: A neuron that learns spike timing-based decisions. *Nat. Neurosci.* **2006**, *9*, 420–428. [CrossRef]
31. Yu, Q.; Tang, H.; Tan, K.C.; Yu, H. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* **2014**, *138*, 3–13. [CrossRef]
32. Wang, X.; Hou, Z.G.; Lv, F.; Tan, M.; Wang, Y. Mobile robots' modular navigation controller using spiking neural networks. *Neurocomputing* **2014**, *134*, 230–238. [CrossRef]
33. Barto, A.G.; Sutton, R.S.; Anderson, C.W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **1983**, *SMC-13*, 834–846. [CrossRef]
34. Masquelier, T.; Guyonneau, R.; Thorpe, S.J. Spike Timing Dependent Plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE* **2008**, *3*, e1377. [CrossRef]