Received: 3 June 2019
DOI: 10.1002/mma.6241

SPECIAL ISSUE PAPER

WILEY

Solving a classification task by spiking neural network with STDP based on rate and temporal input encoding

Alexander Sboev[1,2] | **Alexey Serenko**[1] | **Roman Rybka**[1] | **Danila Vlasov**[1,2]

[1]National Research Centre "Kurchatov Institute", Moscow, Russia

[2]MEPhI National Research Nuclear University, Moscow, Russia

Correspondence
Alexander Sboev, National Research Centre "Kurchatov Institute", Moscow, Russia; or MEPhI National Research Nuclear University, Moscow, Russia.
Email: Sboev_AG@nrcki.ru

Communicated by: T.E. Simos

Funding information
Russian Science Foundation, Grant/Award Number: 17-71-20111

This paper develops local learning algorithms to solve a classification task with the help of biologically inspired mathematical models of spiking neural networks involving the mechanism of spike-timing-dependent plasticity (STDP). The advantages of the models are their simplicity and, hence, the potential ability to be hardware-implemented in low-energy-consuming biomorphic computing devices. The methods developed are based on two key effects observed in neurons with STDP: mean firing rate stabilization and memorizing repeating spike patterns. As the result, two algorithms to solve a classification task with a spiking neural network are proposed: the first based on rate encoding of the input data and the second based on temporal encoding. The accuracy of the algorithms is tested on the benchmark classification tasks of Fisher's Iris and Wisconsin breast cancer, with several combinations of input data normalization and preprocessing. The respective accuracies are 99% and 94% by F1-score.

KEYWORDS

classification task, genetic algorithm, spike-timing-dependent plasticity, spiking neural network learning, synaptic plasticity

MSC CLASSIFICATION

68T05; 92B20

1 | INTRODUCTION

The relevance of building learning models of biologically motivated spiking neural networks is due to the possibility of utilizing these models within neuromorphic computing devices[1-3] with ultra-low power consumption[4] to solve practical classification tasks. Such utilization requires devising three components: encoding the input data into the spike patterns to present to the network, the plasticity mechanism under which the network learns to recognize its input patterns, and decoding the network output spiking activity into the class labels. The existing input encoding methods can be generally divided[5] into rate or temporal encoding, where information is conveyed either by the average number of spikes over some time or by the exact spike timings. Rate encoding has the advantage that an analogy can be drawn between the activation function of a spiking neuron (in terms of the relation of the input rates to the output rate) and that of a formal neuron, leading to the opportunity to map a trained formal network to an identical spiking one.[6] Temporal encoding, in its turn, requires fewer number of spikes, which allows faster computing.[7]

Abbreviations: STDP, spike-timing-dependent plasticity; SNN, spiking neural networks; GRF, Gaussian receptive fields; LIF, leaky integrate-and-fire neuron; IF, integrate-and-fire neuron

For both encoding approaches, rate[1,8,9] and temporal,[10,11] there exist a number of well-performing algorithms to solve pattern recognition tasks. State-of-the-art accuracies on both image recognition and real vector classification were achieved by adapting error backpropagation (BP) to one-layer[12] or multilayer[13,14] spiking networks. BP-based approaches, however, make learning hard to implement directly on the chip. Another notable direction is liquid state networks, where an input pattern is processed by a large reservoir of neurons with random recurrent connections (static or plastic[15]), and within the reservoir, a desired transformation of the input patterns emerges, which has then to be extracted by a readout layer. The readout layer can be trained by conventional optimization algorithms[16] or created by means of structural plasticity.[17] In the latter case, the more patterns are at input, the more neurons are created, which is also hardly scalable for hardware implementation. At the same time, various modifications of Hebbian synaptic plasticity were developed, such as the reward-modulated spike-timing–dependent plasticity (STDP),[18,19] where the amplitude of the weight change is modulated by some global reward signal. In the tempotron[20,21] and BP-STDP[22] learning rules, the plasticity is switched from Hebbian to anti-Hebbian and back again depending on the data sample being inputted; these rules allow their authors to derive target functions that their algorithms perform a gradient descent on.

Generally, in all those learning algorithms, the change of a synaptic weight is governed by some teacher signal that is computed on the basis of the current network input and/or output. This would complicate their implementation in memristive devices, which favor local plasticity rules.[23] This shifts the spotlight to STDP,[24,25] a bio-inspired long-term synaptic plasticity model in which the synaptic weight change depends solely on the timings of presynaptic and postsynaptic spikes. STDP has been shown[26] to be implementable in memristive devices.

For rate encoding, the ability of STDP to perform unsupervised recognition was shown[8]: a layer of excitatory neurons, competing with one another via an additional layer of nontrainable inhibitory neurons, learned to recognize MNIST handwritten digits to the accuracy of 83% with a 100-neuron network, and 87%, 92%, and 95% with 400, 1600, and 6400 neurons, respectively.

For temporal encoding, a promising effect is the ability of a neuron with STDP to memorize repeating spike patterns[27]: when repeating patterns are presented to the neuron alternated with noise, the neuron spikes earlier and earlier in response to the patterns but gradually stops firing spikes during the periods of presenting noise. This effect was previously utilized to distinguish faces from motorbikes[28] and later to recognize MNIST digits.[29]

This paper, striving for the feasibility of hardware implementation, proposes STDP-based learning algorithms for both rate and temporal input encoding with as simple neuron model and network topologies as possible. Our algorithm for rate encoding works with a neuron model simpler than in the existing works and, even more, with far fewer neurons: the number of neurons used just equals the number of classes. For temporal encoding, we improve the pattern-memorizing–based approach by stimulating a neuron to learn patterns of its own class.

The proposed algorithm for rate encoding is based on a robust STDP property of mean spiking rate stabilization,[30] which holds for several neuron models[31] and in a broad range of STDP parameters[32]: STDP leads to such synaptic weights that the neuron fires spikes with a certain mean rate. This rate does not depend on the input spike rates (in a sufficiently wide range) and depends only on the neuron and STDP parameters. In Section 2.5, we utilize this effect for a classification task by training each neuron on its own class and then making it distinguish its own class from the others. Our algorithm for temporal encoding, described in Section 2.4, involves stimulating a neuron to fire a spike early at the beginning of presenting a pattern from the neuron's own class.

In Section 3, we compare the classification accuracy of the two proposed algorithms in combination with different preprocessing techniques (described in Section 2.2) on two real-valued benchmark classification tasks described in Section 2.6: Fisher's Iris[33] and Wisconsin breast cancer.[34] The results in Section 4 show that the learning efficiency is similar for both algorithms provided that a proper input preprocessing is employed.

2 | METHODS

For each of the two ways of input encoding and the two corresponding features of STDP, we describe the corresponding network topology, learning algorithm,* and the method of decoding the network output into the classification result. In addition, we try different techniques of preprocessing the input data. The constants of the neuron

*The code to reproduce learning with temporal encoding is at http://github.com/dartl0l/spiking_network_learning_algorithm. The code to reproduce learning with rate encoding is at http://github.com/sag111/ga_for_snn

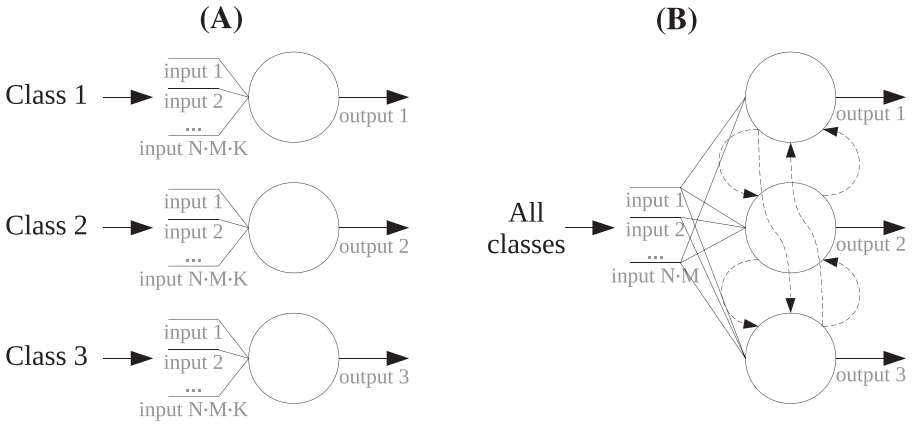

FIGURE 1 A, The scheme of presenting input samples to neurons for training with rate encoding. The number of the neurons equals the number of classes (three here for instance), and each neuron (denoted by a circle) independently receives its own input spike trains encoding vectors of the neuron's own class. The number of input synapses is generally $N \cdot M \cdot K$, where N is the original dimension of the input vector, M is the number of Gaussian receptive fields if applied, and K is the number of additional input multiplication if applied. B, The topology of the network used with temporal encoding. Dashed arrows indicate all neurons having nonplastic inhibitory synapses between each other. All the neurons receive the same $N \cdot M$ input spike trains. Additional K-times input replication is not used

and STDP models, as well as encoding parameters, are optimized for each setup with a genetic algorithm described in Appendix A.

2.1 | Network configuration

For both encodings, the network is one-layer and consists of as many neurons as there are classes, so each neuron is assigned a class (further referred to as the neuron's own class). The difference is, for the temporal encoding, the neurons are connected to each other with nonplastic inhibitory synapses (see Figure 1B) (we admit such topology is less biologically plausible than a dedicated layer of inhibitory neurons,[8] but we omit the latter for the sake of simplicity). For the rate encoding, the neurons are not connected to each other (see Figure 1A).

In both cases, the network is composed of leaky integrate-and-fire (LIF) neurons, every neuron having as many STDP-plastic input synapses as many components an input vector has after passing it through all the preprocessing steps (selected from those listed in Section 2.2).

2.1.1 | Neuron model

The integrate-and-fire (IF) neuron was chosen for both encoding approaches because of its computational simplicity and yet its ability to exhibit, when equipped with STDP synapses, the phenomena necessary for the learning algorithms considered: rate stabilization and memorizing repeating patterns. We consider two variants of the IF neuron: leaky[35] and non-leaky.

The comparison of different input preprocessing combinations for both rate and temporal encoding is performed with the LIF model, in which the membrane potential V obeys

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

and as soon as $V \geq V_{\text{th}}$, V is instantaneously reset back to V_{rest}, and the neuron is unable to change its potential during the refractory period τ_{ref}. We consider the postsynaptic current I_{syn} of exponential form

$$I_{\text{syn}} = \sum_{\text{over synapses } i} \sum_{\text{over spike times } t^i_{\text{sp}} \text{ at } i\text{th synapse}} w_i \left(t^i_{\text{sp}}\right) \frac{q_{\text{syn}}}{\tau_{\text{syn}}} e^{-\frac{t - t^i_{\text{sp}}}{\tau_{\text{syn}}}} \Theta\left(t - t^i_{\text{sp}}\right),$$

where $\tau_{\text{syn}} = 5$ ms, $q_{\text{syn}} = 5$ fC, w_i is the weight of the synapse, and Θ is the Heaviside step function. The neuron constants V_{rest}, V_{th}, τ_{m}, and C_{m} are adjusted with the help of the genetic algorithm, the adjustment ranges and the values found are presented in Table 2. The parameter adjustment ranges, as well as the fixed values (where stated) in the table, are due to our preliminary findings,[36] where parameter search was conducted independently for rate and temporal encoding.

In order to show that the model can be simplified even further, we also try the non-leaky IF neuron model with no refractoriness ($\tau_{\text{ref}} = 0$), and with the postsynaptic current of the delta-function form:

$$\frac{dV}{dt} = \sum_{\text{over synapses}i} \sum_{\text{over spike times}t^i_{\text{sp}}\text{at}i\text{th synapse}} w_i \left(t^i_{\text{sp}}\right) \frac{q_{\text{syn}}}{\tau_{\text{syn}}} \delta\left(t - t^i_{\text{sp}}\right).$$

The parameters to adjust are the same as above.

2.1.2 | Synapse model

The input synapses of the network possess the STDP, in which a synaptic weight $0 \le w \le w_{\max} = 1$ changes by Δw according to the relative timing of presynaptic spikes t_{pre} and postsynaptic spikes t_{post}:

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

We use the additive STDP, with $\mu_+ = \mu_- = 0$, in which case the additional constraint is needed to prevent the weight from falling below zero or exceeding the maximum value of 1:

$$\text{if} \quad w + \Delta w > w_{\max}, \text{then} \ \ \Delta w = w_{\max} - w; \quad \text{if} \quad w + \Delta w < 0, \quad \text{then} \quad \Delta w = -w.$$

Here, the learning rate λ is set as low as the computing resources permit: we set it to 10^{-3}. The remaining parameters α, τ_-, and τ_+ are adjustable, with their adjustment ranges and values found listed in Table 2.

An important part of the STDP model is the scheme[37] of which spikes to take into account as t_{pre} and t_{post} in the rule (1). For the temporal encoding, we use the most common all-to-all spike pairing scheme, where each postsynaptic spike is taken into account in the rule (1) with all preceding presynaptic spikes, and each presynaptic spike is paired with all preceding presynaptic ones. For the rate encoding, we use the restricted symmetric nearest-neighbour scheme, because this scheme was previously shown to exhibit the phenomenon to be utilized—the output rate stabilizing effect. In the restricted symmetric scheme, a postsynaptic spike t_{post} is paired with the latest preceding presynaptic spike t_{pre}, but only if there are no other postsynaptic nor presynaptic (from the same input) spikes between t_{pre} and t_{post}; similarly, a presynaptic spike is paired with the nearest preceding postsynaptic spike if there are no other spikes between.

2.2 | Input preprocessing

We have tried various combinations of the preprocessing methods listed below, subsequently passing the preprocessed vectors to one of the two encoding approaches.

2.2.1 | Minmaxscale

Each component x^i of an input vector $\vec{x}$ is rescaled to the range $[0; 1]$: $x^i \leftarrow \frac{x^i - X^i_{\min}}{X^i_{\max} - X^i_{\min}}$, where $X^i_{\min}$ denotes the minimum of the ith component over all vectors of the training set X: $X^i_{\min} = \min_{\vec{u} \in X} u^i$, and accordingly, $X^i_{\max} = \max_{\vec{u} \in X} u^i$. One might expect such preprocessing to be helpful in such tasks as Fisher's Iris and Wisconsin breast cancer, that have real-valued, and possibly of different scale, input features.

2.2.2 | L2

Each vector is scaled so that its Euclidian norm equals 1. The meaning of this preprocessing for a spiking network with rate encoding is that each input vector feeds the network with the same total number of spikes over all input synapses.

2.2.3 | Gaussian receptive fields

An input vector $\vec{x}$ of the dataset X, originally N-dimensional, is transformed to a vector of a higher dimension $N \cdot M$

$$\left\{g(x^1, \mu_0), \ \dots \ , g(x^1, \mu_M), \ \dots \ , g(x^N, \mu_0), \ \dots \ , g(x^N, \mu_M)\right\}.$$

Each component x^i is expanded into M components $g(x^i, \mu_0), \ \dots \ , g(x^i, \mu_M)$. Here, $\mu_j = X^i_{\min} + (X^i_{\max} - X^i_{\min}) \cdot \frac{j}{M-1}$ is the center of the jth receptive field, where $X^i_{\max}$ and $X^i_{\min}$ are, as above, the maximum and minimum values of ith component over all vectors of the dataset.

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

This preprocessing is a smoothed type of population encoding, commonly used in spiking networks.[20,21,38]

2.3 | Input encoding

In temporal encoding, ith input synapse corresponding to component x^i of a preprocessed input vector $\vec{x}$ receives a spike at the moment $T \cdot (X^i_{\max} - x^i)$. The subtraction from the maximum value $X^i_{\max}$ of x^i over all vectors $\vec{x}$ is introduced here so that the higher x^i is, the earlier the ith input spike is. We use temporal encoding always along with preprocessing by Gaussian receptive fields.

In rate encoding, the input component x^i encoded by a Poisson spike sequence (of length T) of the corresponding mean rate presented to the ith input synapse of the neuron. The transformation of an input vector component to the input spiking rate is linear: x^i is mapped to $\nu_{\text{low}} + x^i \cdot \nu_{\text{high}}$, so that the x^i value of 0 is encoded by a Poisson spike train of rate ν_{low} and the value of 1 is encoded by the rate $\nu_{\text{low}} + \nu_{\text{high}}$. In some cases, multiplication of inputs is performed, so that the ith component of the preprocessed input vector is encoded by a bunch of K inputs receiving independent Poisson trains of the same rate $\nu_{\text{low}} + x^i \cdot \nu_{\text{high}}$. Here, ν_{low}, ν_{high}, and K are adjustable parameters; their adjustment ranges and values found are presented in Table 2. The K-times input duplication is supposed to counter the property[25] of additive STDP to bring weights to a bimodal distribution, where each weight tends either to 0 or to its maximum value of 1. Encoding an input component by a bunch of same-rate inputs makes the sum of the weights of said bunch be the effective weight of the component, thus allowing the weight of an input component to effectively be able to take intermediate values between 0 and 1.

2.4 | Learning algorithm and output decoding for temporal encoding

At initialization, the weights are set to their maximum value of 1. All neurons, fully interconnected with nonplastic inhibitory synapses, receive the same input spike trains. During presenting each training sample, the neuron corresponding to the sample's class is stimulated by a current, which causes the neuron to fire an output spike earlier. This is to facilitate memorizing the pattern encoding sample. Indeed, due to STDP, such an early spike causes rewarding of the synapses that receive early input spikes (which belong to receptive fields corresponding to the current input vector components) and penalizing of the synapses that receive later input spikes. This should eventually lead to facilitation of synaptic weights that cause early output spikes in response to the neuron's own class samples. At the same time, this stimulation-induced spike suppresses spikes of other neurons via inhibitory interconnections, which is supposed to discourage memorization of that sample by other neurons. Besides spike patterns encoding data samples, all inputs receive noise: Poisson spike sequences with low constant frequency.

During the testing stage, both the stimulation with current and the Poisson noise are disabled. The class of a testing sample is determined by which neuron emits the first spike during the presentation of the sample.

2.4.1 | Parameter requirements of the learning algorithm for temporal encoding

This first-spike decoding rule implies that only those input synapses survive to which the own-class patterns feed the earliest spikes, while weights of all the remaining synapses are eventually depressed. Hence, the following considerations for parameter adjustment:

1. The membrane capacity C_{m}, threshold V_{th}, and resting potential V_{rest} should accommodate the number of inputs, each input receiving one spike during the presentation of a pattern.
2. The neuron refractory period τ_{ref} should be equal or higher than the pattern presenting time, so that only one output spike is fired during the presentation of an input pattern. That way, when a pattern from a neuron's own class is

presented, the stimulation-induced early output spike leads to strengthening the earliest input spikes, while a long refractory period ensures that no other output spikes occur and so no other inputs are facilitated.

3. The STDP parameters should be shifted towards more depression in order to discourage weight facilitation as far as possible when a neuron is presented a pattern not from its own class. The depression time constant τ_- should be higher than the facilitation time constant τ_+: τ_+ should be comparable with the pattern presenting time, while τ_- should embrace the period of Poisson noise presented in between input patterns. That way, all the input spikes that come later than an output spike, including the inter-pattern Poisson noise, are accounted for in the STDP depression rule and weaken the synapses they come at.

The above considerations allow one to choose the neuron and STDP parameters based just on the pattern presenting duration T and the inter-pattern interval τ_h: $\tau_{ref} = \tau_h - T$, $\tau_m = T$, $\tau_+ = T$, $\tau_- = \tau_h + T$. Fixing arbitrarily $V_{th} = 1$ and $V_{rest} = 0$, only C_m is left for adjustment. The latter, however, has more effect on the testing stage than on the training stage where spikes are ensured by the stimulation.

2.5 | Learning algorithm and output decoding for rate encoding

The neurons of the network are not connected to each other. In the training stage, each neuron receives only instances belonging to its own class. Then, a neuron is supposed to establish a stable mean output rate in response to the vectors it is trained on. That rate would then be used as the marker to distinguish the neuron's own class.

Before training, weights are initialized with random values uniformly distributed from 0 to 1. We stop the training when each weight gets either below 0.1 or above 0.9, considering it the criterion that the weights have achieved their stationary bimodal[25] distribution, and their further random jitter will not have significant effect on the output spiking rate. In the testing stage, STDP is disabled (technically, by setting $\lambda = 0$), and all neurons are presented with instances of the training and testing sets of all classes.

An instance x from the testing set is assigned to class $k = \text{argmin}|\text{out}_k(x) - \text{out}_k(\text{Class}_k)|$, where $\text{out}_k(x)$ is the spiking rate of the neuron k in response to the input vector x, and $\text{out}_k(\text{Class}_k)$ denotes the mean rate of the neuron k over all instances of the training set of class k. We further call it own-rate decoding rule.

2.5.1 | Parameter requirements of the learning algorithm for rate encoding

Successful learning with the algorithm proposed above for rate encoding depends on the following conditions:

1. The output rate stabilization effect should hold for the given STDP constants, neuron model parameters, and the given scale of input spike rates. This can be achieved by adjusting STDP and neuron constants.
2. For STDP to establish synaptic weights with which the neuron would emit similar mean output rates in response to all samples from the neuron's own class, such weights should at least exist. This requires proper input normalization and/or preprocessing.
3. For a trained neuron in the testing stage to distinguish its own class from the other classes, those other classes should cause it to fire with some other mean rate, different from the neuron's own-class stable firing rate. We achieve this by adjusting the input encoding parameters and by looking for optimal input preprocessing.

2.6 | Datasets

2.6.1 | Fisher's Iris

The dataset[†] available in the UCI repository[39] consists of 150 flowers, described in four features—length and width of sepal and petal in centimeters. The flowers are divided into three classes, 50 in each: *Iris setosa canadenisis*, *Iris virginica*, and *Iris versicolor*. The first class is linearly separable from the second and the third, while the second and the third are not linearly separable.

2.6.2 | Wisconsin breast cancer

The breast cancer dataset[‡] collected in Wisconsin University consists of 569 samples, 357 of "benign" class and 212 "malignant". Features are computed from a digitized image of a fine needle aspirate of a breast mass. The cell nuclei in the image

[†]https://archive.ics.uci.edu/ml/datasets/Iris
[‡]https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

are characterized by ten features: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The input vector consists of the mean, standard deviation, and worst values over all nuclei for each of the features, hence the total of 30 features.

3 | EXPERIMENTS

The following experiments have been conducted to construct an efficient classification algorithm that includes data preprocessing, encoding, and learning and decoding procedures.

Different combinations of normalization options described in Section 2.2—L2, minmaxscale, and L2 with subsequent minmaxscale—have been considered, optionally followed by processing with Gaussian receptive fields. The data, preprocessed in one or another way, are then fed to one of the learning algorithms, the one for rate encoding or the one for temporal encoding.

In order to assess separately the effects of different data representations—the initial raw vectors, or vectors preprocessed with methods mentioned above, or data processed by a trained spiking network—in equal decoding conditions, we compare the accuracy of the Gradient Boosting classifier[§] from the scikit-learn[40] library on the initial raw data and on data preprocessed in various ways, including receptive fields. Moreover, in order to assess the efficiency of the own-rate decoding rule developed in Section 2.5 for the rate-encoding SNN, it is compared with decoding by Gradient Boosting (the latter is trained on the output spiking rates the trained SNN emits in response to the training set and then is to predict the classes of testing set instances by the SNN's output rates in response to them). Concerning the temporal-encoding SNN, its first-spike encoding rule seems straightforward enough, for which reason the experiments with Gradient Boosting are only useful in terms of comparing with initial data representation.

Adjustment of the SNN parameters—neuron and STDP model constants, as well as input encoding parameters—is performed by the MultiNEAT[41] genetic algorithm described in Appendix A. Learning involves fivefold cross-validation. In addition, in order to check the robustness of the model to random initial synaptic weights, we rerun the learning for the best input preprocessing combinations five times and see the accuracy difference from run to run be less than its deviation over cross-validation folds.

After the optimal preprocessing has been found, we set it fixed and study the influence of the SNN model constants. In order to find out whether there exist neuron and STDP parameters common for both learning algorithms and both classification tasks, we set them fixed rather than adjust them genetically: for temporal encoding, we choose them in accordance with the considerations in Section 2.4.1, and for rate encoding, we set the neuron constants equal to those for temporal encoding.

Finally, in order to show that the neuron model can be simplified even further without significant accuracy loss, we run the parameter adjustment and learning anew for the best input preprocessing configuration, replacing the LIF neuron model with the non-leaky integrator described in Section 2.1.1.

4 | RESULTS AND DISCUSSION

Gradient Boosting, when trained either on raw data or on preprocessed data (see "Gradient Boosting" in the column "Learning" in Tables B1 and B2), shows quite the same accuracy regardless of the encoding and preprocessing techniques. This shows that the data remain informative and learnable regardless of preprocessing.

Concerning spiking networks, however, processing with Gaussian receptive fields is a necessary requirement for high accuracy. Spiking network with temporal encoding achieves its best accuracy (see "SNN with temporal encoding" in Table 1) with L2 normalization in the Iris task and with minmaxscale in the breast cancer task. Similarly, spiking network with rate encoding achieves its best accuracy if the data, for both tasks, are minmaxscale-normalized and preprocessed with receptive fields. The preprocessing combination with which both SNN algorithms produce reasonable overall accuracy on both classification tasks is L2 with subsequent minmaxscale followed by receptive fields.

With optimal preprocessing, the "own-rate" decoding rule developed for rate-encoding SNN achieves the same accuracy as decoding by Gradient Boosting (see "Gradient Boosting" in the column "Decoding" of Tables 1, B1, and B2). This means that our decoding rule for rate decoding can accurately extract the information a trained spiking network conveys in its output.

[§]https://scikit-learn.org/stable/modules/ensemble.html$\backslash$#classification

TABLE 1 Accuracy of the best input preprocessing for each learning method

Input Preprocessing	Learning Method	Decoding	F1-Score
Fisher's Iris dataset			
L2 + GRF	SNN with temporal encoding	First-spike rule	99 ± 2
L2 + GRF	SNN with temporal encoding, neuron constants fixed	First-spike rule	97 ± 2
minmaxscale + GRF	SNN with rate encoding	Own-rate rule	97 ± 3
minmaxscale + GRF	SNN of IF neurons with rate encoding	Own-rate rule	94 ± 4
minmaxscale + GRF	SNN with rate encoding	Gradient Boosting	92 ± 5
minmaxscale + GRF	SNN of IF neurons with rate encoding	Gradient Boosting	93 ± 6
minmaxscale + GRF	SNN with rate encoding, neuron constants fixed	Own-rate rule	94 ± 3
	Four-layer formal NN[42]		100
	Insect-inspired CNN-based model[16]		93.33
	SpikeProp[14]		96.1
	BP-STDP[22]		96
Wisconsin breast cancer dataset			
minmaxscale + GRF	SNN with temporal encoding	First-spike rule	89 ± 1
minmaxscale + GRF	SNN with temporal encoding, neuron constants fixed	First-spike rule	90 ± 2
minmaxscale + GRF	SNN with rate encoding	Own-rate rule	90 ± 1
minmaxscale + GRF	SNN of IF neurons with rate encoding	Own-rate rule	94 ± 3
minmaxscale + GRF	SNN with rate encoding	Gradient Boosting	94 ± 1
minmaxscale + GRF	SNN of IF neurons with rate encoding	Gradient Boosting	93 ± 2
minmaxscale + GRF	SNN with rate encoding, neuron constants fixed	Own-rate rule	93 ± 4
	Three-layer formal NN[43]		96
	Insect-inspired CNN-based SNN[16]		96.84
	SpikeProp[14]		97.0
	BP-trained one-layer SNN[12]		99.26

Abbreviations: BP, backpropagation; CNN, convolutional neural network; GRF, Gaussian receptive fields; IF, integrate-and-fire neuron; NN, neural network; SNN, spiking neural network; STDP, spike-timing–dependent plasticity.

TABLE 2 The parameters of SNNs for temporal and rate encoding found for Fisher's Iris and WBC datasets

	Temporal Encoding					Rate Encoding				
		Set 1		Set 2			Set 1		Set 2	
	Adjustment	Values Found for		Values Found for		Adjustment	Values Found for		Values Found for	
Parameter	Range	Iris	WBC	Iris	WBC	Range	Iris	WBC	Iris	WBC
α	[0.3; 1.5]	0.34	0.61	Fixed at 0.65		[0.3; 2]	1.64	1.10	Fixed at 1.8	
λ	[0.05; 0.1]	0.09	0.07	Fixed at 0.03			Fixed at 0.001			
τ_+, ms	[1; 25]	7.18	5.81	Fixed at 6	Fixed at 4	[1; 100]	90	76	Fixed at 70	
τ_-, ms	[5; 40]	37.60	22.36	Fixed at 31	Fixed at 29	[1; 100]	60	36	Fixed at 90	
V_{rest}, mV		Fixed at 0				Fixed at −70			Fixed at 0	
V_{th}, mV		Fixed at 1				Fixed at −54			Fixed at 1	
C_m, pF	[1; 10]	4.16	4.39	Fixed at 1.5		[1; 3]	1.54	1.62	Fixed at 1	
τ_m, ms	[1; 15]	1.83	2.84	Fixed at 6	Fixed at 4		Fixed at 10			
t_{ref}, ms	[1; 15]	7.74	4.65	Fixed at 19	Fixed at 21		Fixed at 3			
M	[5; 30]	20	20	20	10	[2; 25]	7	21	19	20
σ	[0.005; 0.1]	0.005	0.01	0.005	0.01	Fixed at $(X_{max}\text{-}X_{min})/(M\text{-}2)$ as in Yu et al[20]				
T, ms	[1; 6]	6	4	Fixed at 6	Fixed at 4		Fixed at 1 000			
h, ms		Fixed at 25					Not used			
ν_{noise}, Hz	[1; 5]	3.99	5.0	3.20	3.0		Not used			
K		Not used				[1; 25]	24	3	1	
ν_{low}, Hz		Not used				[0; 20]	0.1	0.0	0.0	0.1
ν_{high}, Hz		Not used				[50; 500]	424	218	35	44

Abbreviations: SNN, spiking neural network; WBC, Wisconsin breast cancer.

For comparison, Table 1 also cites a few state-of-the-art results of other existing formal and spiking neural networks. Comparison, however, is complicated by the absence of cross-validation in the literature results.

The SNN parameters obtained by the genetic algorithm are presented for the best preprocessing in "set 1" columns in Table 2. The rate encoding coefficient ν_{high} tends to be high because higher input rates lead to higher output rates, which

make the differences in the neurons' response to different classes more visible. On the contrary, the constant part of the input rate, ν_{low}, is found close to zero.

With the neuron and STDP constants fixed (as presented in "Set 2" column in Table 2) rather than adjusted genetically, the accuracy (rows "neuron constants fixed" in Table 1) is essentially the same as with genetically-adjusted ones, which shows that one can indeed avoid the need to adjust neuron and STDP parameters for each classification task concerned. Adapting to the number and scale of input components in the task concerned can therefore be performed by adjusting the input encoding parameters ν_{low}, ν_{high}, M, and K.

The simplified non-leaky IF neuron model does not impair accuracy compared with the LIF neuron: the results denoted "SNN of IF neurons" in Table 1 differ from the "SNN" results by no more than the deviation range.

5 | CONCLUSION

Both learning algorithms, the first one with rate encoding that is based on rate stabilization effect and the second one with temporal encoding that is based on the pattern memorizing effect, show themselves capable of solving a classification task. With input preprocessing by Gaussian receptive fields, even such a simple model as a one-layer network of IF neurons achieves the F1-score of 99% in the Fisher's Iris dataset and 94% in the Wisconsin breast cancer dataset, which is comparable with the up-to-date accuracies of the multilayer artificial neural networks.

The fact that for both of the learning algorithms two different sets of parameters yield essentially the same accuracy shows that both algorithms are robust to a reasonably wide range of neuron and STDP model constants. The input encoding parameters, however, have to be adjusted for each particular classification task.

The simplicity of the network, along with the additive type of STDP with nearest-neighbour spike pairing scheme that we use, gives a possibility to implement the proposed models in memristive devices. However, further work is needed to ensure that the robustness of the learning algorithm still holds for exactly the plasticity that is observed in such devices.

ACKNOWLEDGEMENTS

Numerical simulations have been carried out with the NEural Simulation Tool (NEST)[44].

This work has been supported by the Russian Science Foundation grant 17-71-20111 and 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/. The authors declare no potential conflict of interests.

ORCID

Alexander Sboev https://orcid.org/0000-0002-6921-4133
Alexey Serenko https://orcid.org/0000-0002-2321-9879

REFERENCES

1. Mitra S, Fusi S, Indiveri G. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *Biomedical Circuits and Systems, IEEE Transactions on*. 2009;3(1):32-42.
2. Pfeil T, Potjans TC, Schrader S, Potjans W, et al. Is a 4-bit synaptic weight resolution enough?—constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in neuroscience*. 2012;6:90.
3. Esser SK, Merolla PA, Arthur JV, Cassidy AS, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*. 2016:201604850.
4. Merolla PA, Arthur JV, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*. 2014;345(6197):668-673.
5. Maass W. Paradigms for computing with spiking neurons. In: van Hemmen JL, Cowan JD, eds. *Models of Neural Networks IV: Early Vision and Attention*. New York, NY: Springer New York; 2002. http://igi-web.tugraz.at/PDF/110mnn4.ps.gz

6. Diehl PU, Neil D, Binas J, Cook M, Liu SC, Pfeiffer Michael. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing; 2015.
7. Maass W. Fast sigmoidal networks via spiking neurons. *Neural Computation*. 1997;9(2):279-304. https://doi=10.1162/neco.1997.9.2.279
8. Diehl PU, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*. 2015.
9. Demin V, Nekhaev D. Recurrent spiking neural network learning based on a competitive maximization of neuronal activity. *Frontiers in Neuroinformatics*. 2018;12:79. https://www.frontiersin.org/article/10.3389/fninf.2018.00079
10. Kasiński A, Ponulak F. Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*. 2006;16:101-113.
11. Gütig R. To spike, or when to spike? *Current Opinion in Neurobiology*. 2014;25:134-139. http://www.sciencedirect.com/science/article/pii/S0959438814000129 Theoretical and computational neuroscience.
12. Ourdighi A, Benyettou A. An efficient spiking neural network approach based on spike response model for breast cancer diagnostic. *Neurocomputing*. 2016;13.
13. Lee JunHaengandDelbruck Tobi and Pfeiffer. *Training Deep Spiking Neural Networks Using Backpropagation*, Vol. 10: Frontiers Media SA; 2016.
14. Bohte SM, Kok JN, La Poutre H. Error-backpropagation in temporally encoded networks of spiking neurons; 2002:17-37.
15. Roy S, Basu A. An online structural plasticity rule for generating better reservoirs. *Neural Computation*. 2016;28(11):2557-2584. http://arxiv.org/pdf/1604.05459.pdf
16. Arena P, Calí M, Patané L, Portera A, Spinosa A. A CNN-based neuromorphic model for classification and decision control. *Nonlinear Dynamics*. 2019;95(3). https://10.1007/s11071-018-4673-4
17. Schliebs S, Kasabov N. Evolving spiking neural networks: a survey. *Evolving Systems*. 2013;4(2):87-98. http://www.zora.uzh.ch/id/eprint/75356/1/Schliebs_Kasabov_Evolving_spiking_neural_networks.pdf
18. Mozafari M, Ganjtabesh M, Nowzari-Dalini A, Thorpe SJ, Masquelier T. Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition. *arXiv preprint arXiv:1804.00227*. 2018.
19. Milad M, Mohammad G, Abbas N-D, et al. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*. 2019;94:87-95. http://www.sciencedirect.com/science/article/pii/S0031320319301906
20. Yu Q, Tang H, Tan KC, Yu H. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing*. 2014;138:3-13.
21. Wang J, Belatreche A, Maguire L, McGinnity TM. An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing*. 2014;144:526-536. http://www.sciencedirect.com/science/article/pii/S0925231214005785
22. Tavanaei A, Maida A. BP-STDP: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing*. 2019;330. http://doi10.1016/j.neucom.2018.11.014
23. Saïghi S, Mayr CG, Serrano-Gotarredona T, et al. Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience*. 2015;9:51.
24. Bi G, Poo M. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience*. 2001;24(1):139-166.
25. Song S, Miller KD, Abbott LF. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*. 2000;3(9):919-926.
26. Serrano-Gotarredona T, Masquelier T, Prodromakis T, Indiveri G, Linares-Barranco B. STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Frontiers in neuroscience*. 2013;7:2.
27. Masquelier T, Guyonneau R, Thorpe SJ. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS One*. 2008;3(1):e1377.
28. Masquelier T, Thorpe SJ. Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology*. 2007;3(2):1-11. https://doi.org/10.1371/journal.pcbi.0030031
29. Kheradpisheh SR, Ganjtabesh M, Thorpe SJ, Timothée M. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*. 2018;99:56-67. http://www.sciencedirect.com/science/article/pii/S0893608017302903
30. Izhikevich EM, Desai NS. Relating STDP to BCM. *Neural computation*. 2003;15(7):1511-1523.
31. Sboev A, Rybka R, Serenko A, Vlasov D, Kudryashov N, Demin V. To the role of the choice of the neuron model in spiking network learning on base of spike-timing-dependent plasticity. In: 8th Annual International Conference on Biologically Inspired Cognitive Architectures Klimov VV, Samsonovich AV, eds., Vol. 123. Elsevier BV; 2018:432-439. https://www.sciencedirect.com/science/article/pii/S187705091830067X
32. Sboev A, Serenko A, Rybka R, Vlasov D, Filchenkov A. Estimation of the influence of spiking neural network parameters on classification accuracy using a genetic algorithm. In: Postproceedings of the 9th Annual International Conference on Biologically Inspired Cognitive Architectures, Vol. 145; 2018:488-494. http://www.sciencedirect.com/science/article/pii/S1877050918323998
33. Fisher RA. The use of multiple measurements in taxonomic problems. II. *Annual Eugenics*. 1936;7:179-188.
34. Street WN, Wolberg WH, Mangasarian OL. Nuclear feature extraction for breast tumor diagnosis. *International Symposium on Electronic Imaging: Science and Technology*. 1993;1905:861-870.
35. Orhan E. The leaky integrate-and-fire neuron model. http://hopf.cns.nyu.edu/~eorhan/notes/lif-neuron.pdf; 2012.
36. Sboev A, Serenko A, Rybka R, Vlasov D. Influence of input encoding on solving a classification task by spiking neural network with STDP. In: Proceedings of the 16th International Conference of Numerical Analysis and Applied Mathematics, Vol. 2116; 2019:270007:1-4.

37. Morrison A, Diesmann M, Gerstner W. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*. 2008;98:459-478.
38. Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions. *Nature Neuroscience*. 2006;9(3):420-428.
39. Dua D, Graff C. UCI Machine Learning Repository. 2017. http://archive.ics.uci.edu/ml
40. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*. 2011;12:2825-2830.
41. Stanley K, Miikkulainen R. MultiNEAT—a portable software library for performing neuroevolution. http://multineat.com/index.html
42. Patil PM, Sontakke TR. Rotation, scale and translation invariant handwritten Devanagari numeral character recognition using general fuzzy neural network. *Pattern Recognition*. 2007;40(7):2110-2117. http://www.sciencedirect.com/science/article/pii/S0031320307000039
43. Murat K, Cevdet Ince M. An expert system for detection of breast cancer based on association rules and neural network. *Expert Systems with Applications*. 2009;36(2, Part 2):3465-3469. http://www.sciencedirect.com/science/article/pii/S0957417408001103
44. Kunkel S, Morrison A, Weidel P, et al. NEST 2.12.0. http://doi=10.5281/zenodo.259534; 2017.
45. Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. *Evolutionary computation*. 2002;10(2):99-127.

How to cite this article: Sboev A, Serenko A, Rybka R, Vlasov D. Solving a classification task by spiking neural network with STDP based on rate and temporal input encoding. *Math Meth Appl Sci*. 2020;1–13. https://doi.org/10.1002/mma.6241

APPENDIX A: THE GENETIC ALGORITHM OF NETWORK PARAMETERS ADJUSTMENT

A.1 | Description of the genetic algorithm

The algorithm we employ to search for optimal spiking network model constants is NeuroEvolution of Augmented Topologies (NEAT)[45] in the implementation of MultiNEAT.[41] We have chosen it because of its ability to adjust the network topology: although we keep the topology fixed throughout this paper, we elected to save the opportunity to look for arbitrary topologies in future work.

MultiNEAT starts with generating a population of parameter sets (called individuals), distributed uniformly over the parameter space. Then, the individuals are evaluated by their fitness. After that, individuals undergo mating, mutations, arranging into species, the latter competing against each other with a chance to become extinct. The fitness function we use is just the F1-macro score on the training set. We had also tried other functions, specified in terms of neurons' output rates, that provided some smoother measures of how good or bad the network output was. However, the best accuracy obtained did not depend on the fitness function used, so the experiments presented are conducted with just the F1 as the fitness.

The genetic algorithm comprises the following steps.

A.2 | Fitness function normalization

After evaluating all individuals by their fitness, the fitness function of each individual is normalized by the number of individuals in its species to penalize overpopulated species. Furthermore, individuals from young species (species created less than five generations ago) get their fitness multiplied by 1.1 in order to encourage mutations that may eventually be useful but reduce fitness in short term. Accordingly, the fitness of species older than 30 generations is multiplied by 0.5. If the total fitness of a species does not improve during 50 generations, it is multiplied by 10^{-7}, which inevitably makes the species extinct in the next generation.

A.3 | Mating

One percent of the best performing individuals enter the next generation unchanged. Twenty-five percent of the best individuals produce offspring, with the probability of 0.3 it happens by asexual reproduction (just copying), otherwise—by

mating with each other, in which case the sibling's parameters are an average of the parents' ones. Interspecies mating occurs with probability of 10^{-4}. During mating, a mutation occurs with probability of 0.25. Each adjustable parameter value has the probability of 0.1 to be involved in the mutation; being mutated, a parameter is chosen anew from a uniform distribution (with minimum and maximum boundaries as in "adjustment range" column of Table 2) with probability of 0.25%, otherwise changed by a random value uniformly distributed.

A.4 | Speciation

Then, all individuals are rearranged into species. An individual is assigned its species by calculating for each species the distance between a random representative of the species (from the previous generation) and the individual. If the distance to some species does not exceed some internal variable CompatibilityThreshold, the individual is assigned to that species (without further considering any other species, so the species assigning algorithm is of a first-fit kind). CompatibilityThreshold changes dynamically so that there always be 5 to 10 species. If an individual is farther than CompatibilityThreshold from all species, a new species is created for it. If after the rearrangement a species loses all its individuals, it becomes extinct.

APPENDIX B: ACCURACIES OF ALL INPUT PREPROCESSING COMBINATIONS

TABLE B1 Accuracy on the Fisher's Iris dataset

Input Preprocessing		Learning Method	Decoding	F1-Score			
Normalization	**GRF**			**Mean**	**Std**	**Min**	**Max**
–	**–**	**Gradient Boosting**		**96**	**5**	**87**	**100**
L2	–	Gradient Boosting		95	3	90	100
L2	–	SNN with rate encoding	Own-rate rule	90	8	75	100
L2	–	SNN with rate encoding	Gradient Boosting	92	7	82	100
minmaxscale	–	Gradient Boosting		95	6	83	100
minmaxscale	–	SNN with rate encoding	Own-rate rule	88	7	80	97
minmaxscale	–	SNN with rate encoding	Gradient Boosting	88	5	83	93
L2 + minmaxscale	–	Gradient Boosting		92	5	82	97
L2 + minmaxscale	–	SNN with rate encoding	Own-rate rule	88	7	82	97
L2 + minmaxscale	–	SNN with rate encoding	Gradient Boosting	93	5	86	100
L2	+	Gradient Boosting		95	2	93	97
L2	+	SNN with rate encoding	Own-rate rule	93	5	87	100
L2	+	SNN with rate encoding	Gradient Boosting	95	2	93	97
L2	+	SNN with temporal encoding	First-spike rule	99	2	97	100
minmaxscale	+	Gradient Boosting		95	3	90	100
minmaxscale	+	SNN with rate encoding	Own-rate rule	97	3	93	100
minmaxscale	+	SNN with rate encoding	Gradient Boosting	92	5	86	100
minmaxscale	+	SNN with temporal encoding	First-spike rule	83	3	77	87
L2 + minmaxscale	+	Gradient Boosting		91	3	87	97
L2 + minmaxscale	+	SNN with rate encoding	Own-rate rule	92	5	86	100
L2 + minmaxscale	+	SNN with rate encoding	Gradient Boosting	90	7	83	100
L2 + minmaxscale	+	SNN with temporal encoding	First-spike rule	97	3	93	100

Abbreviations: GRF, Gaussian receptive fields; SNN, spiking neural network.

TABLE B2 Accuracy on the Wisconsin breast cancer dataset

Input Preprocessing		Learning Method	Decoding	F1-Score			
Normalization	**GRF**			**Mean**	**Std**	**Min**	**Max**
–	**–**	**Gradient Boosting**		**95**	**2**	**92**	**98**
L2	–	Gradient Boosting		95	3	90	99
L2	–	SNN with rate encoding	Own-rate rule	88	3	84	91
L2	–	SNN with rate encoding	Gradient Boosting	83	5	77	89
L2 + minmaxscale	–	Gradient Boosting		93	2	91	95
L2 + minmaxscale	–	SNN with rate encoding	Own-rate rule	70	5	64	77
L2 + minmaxscale	–	SNN with rate encoding	Gradient Boosting	67	7	56	75
L2	+	Gradient Boosting		95	3	91	100
L2	+	SNN with rate encoding	Own-rate rule	89	3	84	92
L2	+	SNN with rate encoding	Gradient Boosting	83	5	79	91
L2	+	SNN with temporal encoding	First-spike rule	89	3	84	93
minmaxscale	+	Gradient Boosting		94	1	93	95
minmaxscale	+	SNN with rate encoding	Own-rate rule	90	1	88	92
minmaxscale	+	SNN with rate encoding	Gradient Boosting	94	1	92	94
minmaxscale	+	SNN with temporal encoding	First-spike rule	89	1	88	91
L2 + minmaxscale	+	Gradient Boosting		91	3	88	95
L2 + minmaxscale	+	SNN with rate encoding	Own-rate rule	83	7	72	90
L2 + minmaxscale	+	SNN with rate encoding	Gradient Boosting	87	7	73	93
L2 + minmaxscale	+	SNN with temporal encoding	First-spike rule	88	3	83	93

Abbreviations: GRF, Gaussian receptive fields; SNN, spiking neural network.