Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 462 – 480
© TÜBİTAK
doi:10.55730/1300-0632.3995

SPAYK: an environment for spiking neural network simulation

Aykut Görkem GELEN[1,*], **Ayten ATASOY**[2]
[1]Department of Electrical and Electronics Engineering, Erzincan Binali Yıldırım University, Erzincan, Turkey
[2]Department of Electrical and Electronics Engineering, Karadeniz Technical University, Trabzon, Turkey

Received: 01.08.2022 • **Accepted/Published Online:** 02.03.2023 • **Final Version:** 23.03.2023

Abstract: In research areas such as mobile robotics and computer vision, energy and computational efficiency have become critical. This has greatly increased interest in high-efficiency neuromorphic hardware and spiking neural networks. Because neuromorphic hardware is not yet widely available, spiking neural network studies are conducted by simulations. There are numerous simulators available today, each designed for a specific purpose. In this paper, a novel and open-source package (SPAYK) for simulating spiking neural networks is presented. SPAYK has been proposed to speed up spiking neural network research. In the majority of simulators, networks are expressed with differential equations and require advanced neuroscience knowledge since such simulators are generally designed for brain and neuroscience research. SPAYK, on the other hand, is specifically designed as a framework to easily design spiking neural networks for practical problems. SPAYK is an easy-to-use Python package. There are three fundamental classes in the core: the model class for creating neuron groups, the organization class for simulating tissues, and the learning class for synaptic plasticity. While developing and testing the SPAYK environment, various experiments were carried out. This study includes three of these experiments. In the first experiment, we investigated the behavior of a group of Izhikevich neurons for visual stimuli. Also, a single Izhikevich neuron has been trained to respond to a particular label in a supervised manner with synaptic plasticity. In the second experiment, a well-known experiment was repeated to validate SPAYK. In this experiment, a neuron trained by synaptic plasticity can recognize repetitive patterns in a spike train. In the third experiment, a similar neuron was simulated with stimuli with multiple labels adapted from the MNIST dataset. It has been shown that the neuron can classify a particular label by synaptic plasticity. All these experiments and the SPAYK environment are presented as open-source tools.

Key words: Spiking neural network, STDP based learning, supervised classification, unsupervised pattern recognition

1. Introduction

Artificial neural network research has a wide range of applications and has a significant impact on many different research areas. Studies in the field follow an increasingly complex course such as the development of perceptron, fully connected networks, convolutional neural networks, recurrent networks, and attention mechanisms. Despite the tremendous progress in network architectures, network training has continued to adhere to the principle of backpropagation and gradient based other methods. The backpropagation algorithm based on mathematical foundations has become a standard for artificial neural network training except for some derivative-free optimization methods. This dependency necessitates the use of differentiable blocks in the design of new network architectures.

*Correspondence: aykut.gelen@erzincan.edu.tr

In neural network designs, an evolution toward biological structures has been noticed due to computational and energy efficiency in biological structures. Third-generation spiking neural networks are being developed to exploit these advantages. A spiking neuron, unlike previous generations, attempts to extract information from spike signals, rather than abstract numerical values. Many biological processes, such as the opening and closing of ion channels and the release of neurotransmitters, are mimicked to design a spiked neuron. This approach has resulted in the development of mathematical models for neurons and synapses at various levels of complexity. Today, spiking neural networks are a very active field of research, with significant advancements.

In the brain, the event-driven nature of the signals and the encoding of information in the form of spike signals allow an enormous data processing process to be performed with very little energy. For this reason, neural network research is developing toward approaching biological models. It has been argued for a long time that the future of neural networks will be realized through a spike-based computation model. In spiked neural networks, information is encoded in spike signals. Keeping the information as spike trains allows the information to be expressed in binary form as in standard computer architecture but in a time-dependent manner. This reduces the transmission and processing costs of information. For all these reasons, the computational and energy efficiency of the spiking calculation model is higher than the previous generations.

Aside from their advantages, spiking neural networks have some drawbacks. Spiking neural network training has not been solved as clearly as in previous generations. A gradient-based universal solution, as in artificial and convolutional networks, has yet to be proposed. The first issue is that the loss functions and intermediate model functions are not differentiable. As a result, many simple ideas used in previous-generation networks cannot be applied to spiking networks. Another issue is synapse models. In spiking neural networks the concept of weights are not well defined. Connections between neurons and trainable parameters vary depending on the complexity of the neuron model used. The diversity of the selected synapse and ion channel models requires changes according to the model during the training process. For spiking network training, various approaches have been proposed. These approaches are primarily based on the adaptation of traditional methods or biological processes like Hebbian learning rules and synaptic plasticity.

With spiking neural networks, it is possible to solve the problems that first or second-generation neural networks have effectively overcome. To make classical problems suitable, it is sufficient to convert the input data into spike signals. Abstract numerical values in any given data, for example, can be interpreted as the injected currents or firing rates of neurons. As a result, input data can be mapped directly to spike patterns. The resulting spike patterns can be processed by clusters of spiking neurons connected in various ways. Thus, potential solutions to classical neural network problems can be investigated.

Researchers from various fields are interested in spiking neural networks. Different simulation and experimental environments are used by researchers in fields such as neuroscience, computer science, psychology, and robotics. As a result, some of the available tools are specialized for studying biological processes, while others are specialized for studying neurosicience problems. Some simulation environments use differential equations to express models, while others use a higher-level syntax. Considering this diversity, it is believed that a simulation environment with understandable and simple commands is required, particularly for robotics researchers unfamiliar with neuroscience. In addition to ease of use, runtime and memory usage are also important parameters for simulators. The purpose of use is also decisive for these parameters. While the total executation time of the simulation is very important for a brain region study consisting of many neurons, memory usage and instant executation time may be important for a simulation running on a mobile platform.

1.1. Related work

For a long time, several neural network models and the idea of reducing computational power have been debated, and spiking neural networks, categorized as the third generation, are being developed. In Maass's 1997 review, spiking neural networks are compared to prior generation models in terms of processing capacity [1]. It was demonstrated in the study that a single spiking neuron model can approximate some arbitrary functions that can be approximated by multiple perceptrons or sigmoidal neurons by simulation.

A simulation is a powerful tool for spiking neural networks. Aside from in vitro and in vivo experimental studies, neuron models and network structure simulations are used in many kinds of research. There is a great deal of research for solving practical problems, in addition to modeling parts of the brain or developing neuron models. Many computer vision studies have been carried out with spiking neural networks by simulations. Suggested solutions for energy-efficient object detection, robust object detection, and object tracking problems for robotic control are some of them [2–4]. Classical classification problems, such as EEG signal classification and sound classification, have been studied and solved with spiking neural networks [5, 6].

Many tools and packages have been developed to simulate spiking neural networks. These simulators differ in terms of the areas in which they have been used and how the models are expressed. Tikidji-Hamburyan et al. published a comparative study in 2017 that provides a detailed overview of popular simulators [7]. The methods were investigated in terms of computational complexity, available hardware, and user support. In the study, BRIAN, GENESIS, NEURON, and NEST simulators were investigated in two case studies. As a result, it has been emphasized that BRIAN is superior in terms of expression, NEST in terms of large-scale models, and NEURON in terms of detailed biological models. BRIAN is one of the most widely used tools for simulating spiking neural networks [8]. BRIAN includes intricate models of many neurons and synapses. Models can be simulated by expressing them as differential equations. It is used in studies such as the behavior of groups of neurons and specific brain area models.

Another environment for simulating spiking neural networks is being developed by the Neural Simulation Technology Initiative [9]. Their simulator is called as neural simulation tool (NEST). NEST is another widely used tool in neuroscience and is a tool used and developed under the Human Brain Project (HBP). Thus, NEST is intended to work with large models comprised of many neurons. The main goal is to simulate brain structures and dysfunctions. NEURON, GENESIS, and CARLSim are other important spiking neural network simulators. NEURON was created as a simulation environment and is commonly utilized in neuroscience research. NEURON was designed as a simulation environment and is frequently used by neuroscientists. It is especially emphasized that the models in the NEURON simulator are quite compatible with the experimental data [10]. The NEURON simulator's highly detailed models make the simulations more biologically consistent. Although GENESIS is a very old simulator, it has a significant place in the literature [11]. Because of its modular design and performance, it has been employed in numerous investigations. CARLSim is another spiking neural network simulator with GPU acceleration support. It was developed by the Cognitive Anteater Robotics Lab. and supports a variety of neuron models. There are many versions of CARLSim [12].

Nengo is another significant simulator. It is based on a theory on neural network design for cognition purposes called the Neural Engineering Framework (NEF) [13]. Nengo offers training schemes for spiking neural networks, deep learning approaches, and many models of brain regions [14]. Nengo was created specifically for the study of large-scale brain models. Nengo is used to create many different brain area models and cognitive models with practical applications. Unlike other simulators, the ANNarchy uses a code generation strategy [15]. ANNarchy has been developed especially for rate-coded networks and offers support to simulate on the

GPU. Networks that can be easily written in Python with the code generation mechanism are translated into a low-level language such as C++, and speed advantage is used. In the ANNarchy environment, neuron groups and connections are defined as differential equations as in other simulators.

Apart from these simulators, spiking neural network simulators based on classical convolutional neural network frameworks such as PyTorch and TensorFlow have been developed. The simulator, called SpykeTorch, is based on the PyTorch library and has packages to perform the design and simulation of networks known as deep convolutional spiking neural networks [16]. Similar to the PyTorch library, the convolution operator is applied to the time-varying spike signals to extract features and perform training based on synaptic plasticity. SpikingJelly is another Pytorch-based simulator that can be considered in the same category [17]. It has similar features and can support the conversion of artificial neural networks into spiking networks. Such simulators try to solve traditional neural network problems with spiking neural networks and do not aim for biological consistency.

The simulation of the SNNs is carried out for different purposes such as investigating the functioning of spesific brain regions, synaptic plasticity or hebbian learning rules. A training approach like traditional neural networks is needed to use SNNs for solving practical engineering problems such as computer vision, object recognition, localization or mapping. Although there are some attempts to train SNNs for spesific tasks, unfortunately, it cannot be performed with classical gradient-based training algorithms. Various approaches have been presented for learning in spiking networks. Network training studies for SNNs are usually on the adaptation of classical methods or the application of synaptic plasticity rules. The study published by Lee et al. is one of the important studies on the adaptation of the traditional backpropagation method to spiking neural networks [18]. The solution is obtained by deriving differential equations of neuron models and attempting to replicate the backpropagation algorithm using mathematical tricks. It has been shown in the study that the use of regularization and thresholding can result in a stable learning process. Surrogate gradient learning, proposed by Neftci et al., is another training method with the same approach [19]. Since the activation functions are not differentiable for spiking models, it has been argued that a gradient-based training can be brought to spiking neural networks by using surrogate gradients instead of calculating gradients directly.

Spiking network training techniques can be divided into two categories: supervised and unsupervised. In the review study published by Wang et al., the supervised training approach was grouped according to different perspectives such as network architecture, running mode and information encoding [20]. In the study, supervised training strategies were grouped as perceptron-based algorithms that try to reduce the difference between calculated output and supervised target, synaptic plasticity algorithms based on Hebbian learning rules, and spike train convolution algorithms like convolutional neural networks. Some studies use the famous MNIST dataset for supervised training for the image classification problem of spiked neural networks. In the study published by Li et al., a spiking neural network that tries to imitate V1 and V2 areas inspired by the visual cortex was trained as a supervised and the dataset was classified [21]. In the study, it has been shown that a bioinspired spiking network based on orientation-selective cell models can be trained in a supervised manner. The study also presented a tempotron supervised learning rule based on synaptic plasticity. In the article published by Liu et al. in 2018, a spiking neural network was trained unsupervised to solve an image classification problem using synaptic plasticity [22]. In the study, a hierarchical network inspired by the primary visual cortex structure was designed and trained using spike timing dependent plasticity (STDP). Images from MNIST and CIFAR datasets were used. It has been shown that the training of the network is successful and the results are presented comparatively.

1.2. Contributions

In this paper, we present SPAYK, an open-source environment for spiking neural network simulations. SPAYK has been proposed as an alternative to existing simulation environments. Artificial or convolutional neural network frameworks, which are widely used, have enabled researchers from various disciplines to develop applications without having to understand deep learning or the mathematical details of network training. This has accelerated interdisciplinary research. Our goal in developing SPAYK was to create a framework with a relatively simple syntax that could be used by researchers outside of the neuroscience field. SPAYK aims to enable researchers in robotics and computer vision to create spiking neural networks tailored to practical problems and run them on mobile platforms using traditional CPU or GPU hardware. The main contributions of our paper are listed below.

- SPAYK is offered as an open source python package [§]. All experiments in the article and many experiments in our work are presented together with SPAYK. This ensures that our experiments are reproducible.

- Three different experiments were conducted to demonstrate the use and capabilities of the SPAYK environment. Experiments on learning with spike timing dependent plasticity (STDP) based on Izhikevich neuron groups in an supervised manner and spike response model were carried out. Experiment results and SPAYK environment were verified by repeating the unsupervised pattern recognition study published by Masquelier et al. [23]. The problem of unsupervised classification in the presence of other samples was studied with the MNIST dataset, which is popular in the artificial neural network literature. The third experiment designed for this purpose and results are presented.

- The execution time and memory usage of the SPAYK-supported neuron models were measured and reported for groups with varying numbers of neurons and synapses. The results of the performance analysis demonstrated that the SPAYK implementation is scalable for a large number of neurons and synapses in terms of execution time and memory consumption.

The paper is organized as follows. The SPAYK environment is introduced and the basic modules are detailed in the second section of the paper. The third section contains the designed experiments, the experiment results, and a general discussion. The final section contains the conclusions.

2. SPAYK environment

SPAYK is a Python package that has been designed to describe and simulate spiking neural networks. SPAYK consists of model, organization and learning classes as the core. The model class includes neuron models with different dynamics. Neuron groups can be created using these models. The neuron groups can be stimulated with different inputs such as current sources and Poisson spike trains. In the SPAYK environment, neuron groups can be configured to synaptic plasticity and learning experiments can be performed. The learning class offers support for synaptic plasticity. Tissue functions and the simulator are located in the organization class. The organization class assembles the created architecture as a tissue and enables the features of the simulation core to be used. The following sections describe the background of components.

[§]You can check the repository at https://github.com/aggelen/Spayk to view the source code.

2.1. Neuron models and dynamics

The Hodgkin-Huxley (HH) model is a well-known and generalized way to express the membrane potential as a differential equation system [24]. The model is based on conductances and describes the electrical relationships between the membrane potential and the ion channels. Because it is a sophisticated model with many ion channels, the HH model is particularly difficult to simulate. Approximate models based on various approaches have been proposed to facilitate the simulation of nerve cells and neuron groups. The Izhikevich model and the leaky integrate and fire (LIF) model are two of the most common of these models. In the study presented by Skocik and Long, the capabilities and computational complexities of these three models are compared [25]. The study found that for each millisecond of simulation time, 0.12 s, 1.05 s, and 2.11 s should be spent for the simulation of the LIF, Izhikevich, and HH models, respectively.

SPAYK currently supports these two relatively less complex models. Neuron groups can be defined as having the Izhikevich model with different dynamics or leaky integrate and fire (LIF) model with the Spike response approach. The Leaky integrate and fire model was added to facilitate synaptic connections and weighting to support learning research with the spike response model (SRM).

The Izhikevich model is a popular model that reduces neuron dynamics to a differential equation system. By varying the parameters of the differential equations, different neuron behaviors can be achieved. The Izhikevic spiking neuron model is expressed as a two-dimensional system of differential equations [26]. The model is given by Equations 1 and 2. All parameters are dimensionless variables. The membrane potential and recovery parameters are denoted by v and u, respectively. Parameters a, b, c and d in the differential equation are constants used to change the dynamics of the neuron model. The values of constants for different dynamics are presented in the paper published by Izhikevich [26].

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t} = 0.04v^2 + 5v + 140 - u + I(t) \tag{1}$$

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = a(bv - u) \tag{2}$$

A spike occurs when the membrane potential exceeds a certain threshold. The original article set this dimensionless threshold value to 30. When the spike occurs, the equation system must be reset using the rules specified in Equation 3.

$$v(v > 30) = c \text{ and } u(v > 30) = u + d \tag{3}$$

An Izhikevich neuron group with 100 neurons was simulated for 1000 ms using SPAYK. The neurons in the group have different dynamics. They could behave like any of the cortical neurons described in Izhikevich's paper, including regular spiking, intrinsically bursting, chattering, fast-spiking, thalamocortical, resonator, and low-threshold spiking dynamics [26]. The dynamics of the neurons in the experiment were chosen randomly from this group so that the distribution was uniform. Random constant current values starting and ending at random times were used as stimuli. The current signals are generated with a fixed amplitude of 5 uA, with a random starting time between 100 ms and 400 ms and a random ending time between 600 ms and 900 ms. The membrane potential of 5 random neurons is given in Figure 1. The raster plot can be created to show the firing patterns of all neurons. Figures can be reproduced with the script in the SPAYK repository at `experiments/devel_01_izhikevich_neuron_dynamics.py`.

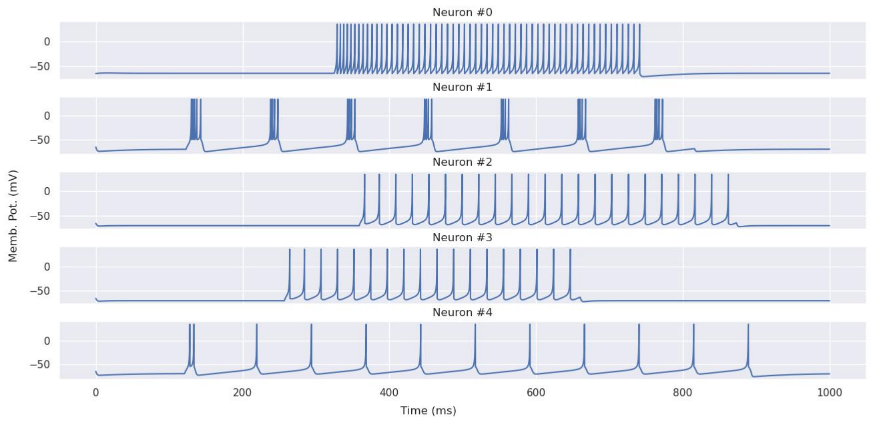

Figure 1. Membrane potential of Izhikevich neurons with different dynamics.

To reduce the complexity, there are simplified models called Integrate and Fire. The LIF model is one of the most popular of these models. Equation 4 expresses the LIF model.

$$\tau_{\mathrm{m}} \frac{\mathrm{d}v(t)}{\mathrm{d}t} = -(v(t) - E_L) + RI(t). \tag{4}$$

The LIF model is represented as a simple electrical circuit consisting of a parallel capacitor and a resistor driven by current $I(t)$ [27]. In the differential equation of the circuit, the membrane potential is represented by $v(t)$. The E_L represents the resting potential and τ_{m} represents the time constant. According to the model, the membrane potential decreases towards the resting potential in the absence of any stimuli. When the membrane potential exceeds a specified threshold value, the neuron fires and the resting stage begins.

Gerstner proposed the spike response model (SRM) by generalizing the LIF model [28]. SRM takes a different approach that is based on filtering rather than a differential equation solution. The model is given by Equation 5.

$$u(t) = \eta(t - \hat{t}) + \int_0^{\infty} \kappa(t - \hat{t}, s) I(t - s) ds \tag{5}$$

The equation consists of two parts. The η function in the first part represents the membrane potential change due to the neuron's dynamics, and the κ part represents the change in the membrane potential due to the stimuli. In the equation, $\hat{t}$ represents the last firing time of the neuron, I represents the injected current. In the model, firing is determined by thresholding the membrane potential. SPAYK supports groups of SRM neurons. A single SRM neuron was simulated for 1000 ms using SPAYK. A Poisson spike train consisting of 10 neurons was used as the stimuli. It is assumed that each neuron in the stimuli fires with random rates, and these rates change over time. The stimuli is shown in Figure 2a. SRM neuron was simulated using this spike train. The plot of the membrane potential of the neuron against time is given in Figure 2b. Figures can be reproduced with the script in the SPAYK repository at `experiments/devel_02_spike_response_model.py`.

2.2. Synapse models

In Spiking neural network simulation, synapse models are classified as current-based (CUBA) or conductance-based (COBA). The synaptic current is calculated as the direct product of the spikes and the weights in the

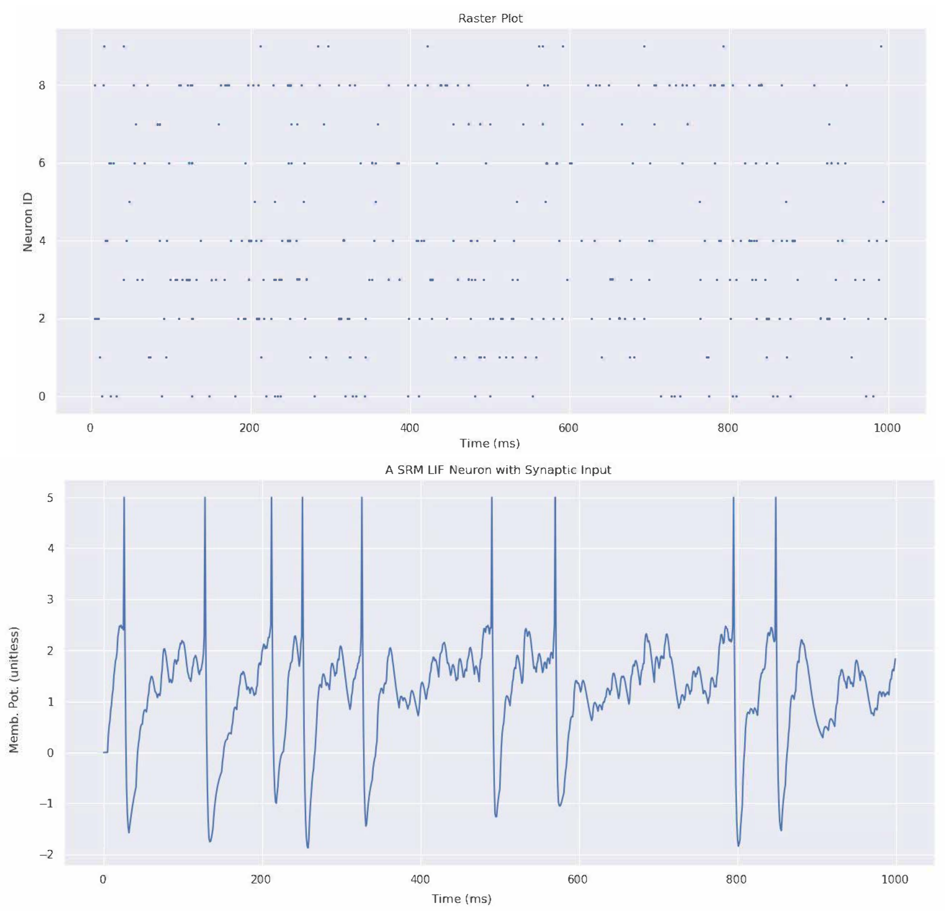

Figure 2. (a) Raster plot of input stimuli, (b) membrane potential of SRM LIF neuron.

current-based synapse model. It simplifies the overall model by simplifying synaptic equations. In the CUBA model, ion channels are modeled as conductances and the total synaptic current is found by calculating the conductances for different neurotransmitters. The channel models for $AMPA$, $NMDA$, $GABA_A$ and $GABA_B$ are defined as a single exponential decay or a combination of two exponential decays [27].

Equations 6 and 7 describe the conductance dynamics that are modeled by a single exponential and a combination of two exponential decays [27]. In the equations, $\bar{g}_{\text{syn}}$ represents a constant peak conductance, $t^{(f)}$ spike time, Θ Heaviside function, τ_{fast} and τ_{slow} represent the time constants of fast and slow exponential decays in the double decay model, respectively, and α represents the mixing ratio between two exponential decays.

$$g_{\text{syn}}(t) = \sum_f \bar{g}_{\text{syn}}\, \mathrm{e}^{-(t-t^{(f)})/\tau}\, \Theta(t-t^{(f)})\,, \tag{6}$$

$$g_{\text{syn}}(t) = \sum_f \bar{g}_{\text{syn}}\, [1-\mathrm{e}^{-(t-t^{(f)})/\tau_{\text{rise}}}] \left[a\, \mathrm{e}^{-(t-t^{(f)})/\tau_{\text{fast}}} + (1-a)\, \mathrm{e}^{-(t-t^{(f)})/\tau_{\text{slow}}}\right] \Theta(t-t^{(f)}). \tag{7}$$

SPAYK supports channel models. The plot of different channel models is given in Figure 3. In the figure, synaptic currents are plotted showing the change in conductance of different channel models for a spike that exists at time zero. While $AMPA$ and $NMDA$ have a positive effect for excitatory connections, $GABA_A$ and $GABA_B$ have a negative effect for inhibitory connections. Figures can be reproduced with the script in the SPAYK repository at `experiments/devel_06_synaptic_channels.py`.

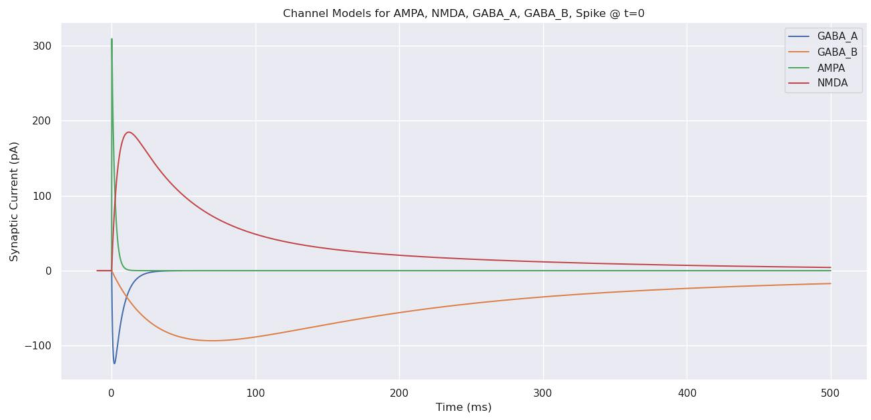

Figure 3. Synaptic channel models.

2.3. Learning with synaptic plasticity

The learning class in the Spayk environment supports an STDP class that can be used offline. In addition, neuron groups in the models class can display STDP behavior online. STDP is a type of plasticity that strengthens or weakens synaptic connections based on spike times.

One of the most important sources on synaptic plasticity is Hebb's 1949 postulate [29]. The principles called Hebb learning rules to form the basis for STDP. According to Hebb learning rules, if neurons are close to each other, fire one after the other and presynaptic neurons take part in the postsynaptic neurons firing, the connection between these neurons becomes stronger. In the opposite case, the connection between them weakens. Hebb's rules have a solid empirical foundation with the long term potentiation (LTP) and long term depression (LTD) studies. STDP is a synaptic plasticity model created from experimental data. The model is expressed by the piecewise function given by Equation 8. In the equation A_+ and A_- express the power of exponentials, τ_+ and τ_- represent the time constants of the STDP window. Presynaptic and postsynaptic spike times are expressed as t_j and t_i, respectively. An example STDP curve plotted for the parameters A_+ 0.03125, A_- 0.028, τ_+ 16.8, τ_- 33.7 is given in Figure 4.

$$\Delta w = \begin{cases} A_+ \exp\left(\frac{t_j - t_i}{\tau_+}\right) & \text{if } x \in t_j \leq t_i \\ -A_- \exp\left(\frac{t_i - t_j}{\tau_-}\right) & \text{if } x \in t_j > t_i \end{cases} \tag{8}$$

2.4. System architecture

SPAYK is composed of model, learning and organization classes. An experiment in SPAYK is carried out by running an organization in the simulator core. It is intended to make it simple to describe the network architecture and to operate with few commands. An organization could be a single neuron or a tissue formed by the connection of multiple neurons with a specific architecture. In the SPAYK environment, a tissue is an

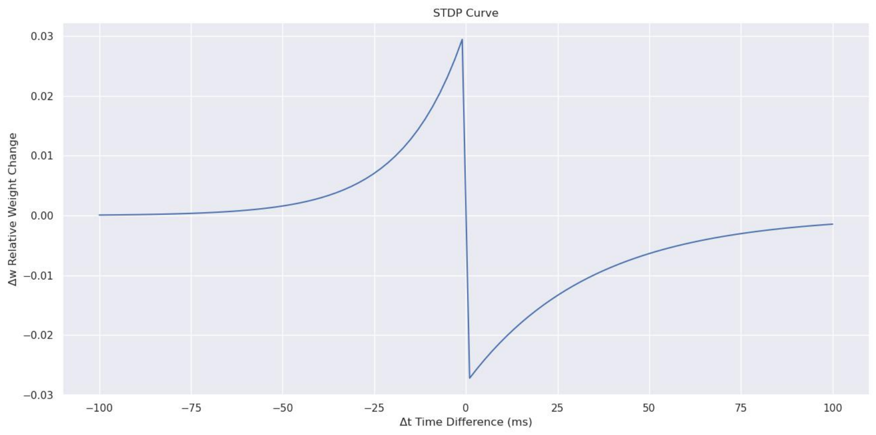

Figure 4. Example STDP curve.

architecture that depicts the neural structure. Afterwards, any tissue can be easily simulated with an input stimulus.

An example script is provided as Listing 1 to illustrate the usage of SPAYK. After the necessary components are imported, an SRM group with 10 neurons is created and connected to a tissue. A Poisson spike train is then simulated for 1000 ms as the stimuli.

Listing 1. Example tissue and core.

```
from spayk.Organization import Tissue
from spayk.Models import SRMLIFNeuron
from spayk.Stimuli import PoissonSpikeTrain

import numpy as np
import matplotlib.pyplot as plt
plt.close('all')

# create two neurons as a group with LIF/SRM model
group_params = {'n_synapses': 10, 'dt': 1.0}

srm_neuron_group = SRMLIFNeuron(group_params)

# bind neuron groups to a tissue
test_tissue = Tissue([srm_neuron_group])
input_spike_train = PoissonSpikeTrain(dt=1.0,
                                      t_stop=1000,
                                      no_neurons=10,
                                      spike_rates=np.random.randint(10,60,10))
input_spike_train.raster_plot()

# run simulation
test_tissue.keep_alive(stimuli=input_spike_train)
test_tissue.logger.plot_v()
```

A simplified unified modeling language (UML) diagram is given in Figure 5 to present the SPAYK classes as a summary. In diagram, main classes are shown with their class names and methods. Inheritance between classes are indicated by blue arrows.

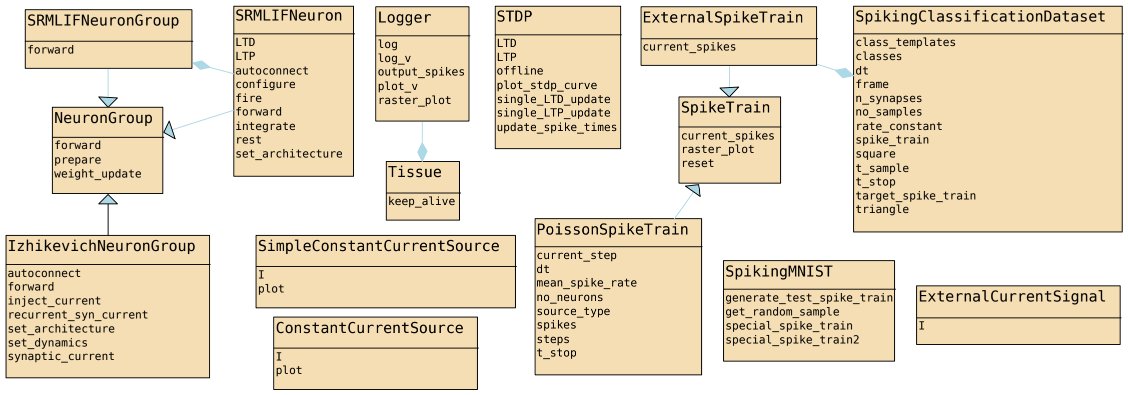

Figure 5. UML diagram of main spayk classes.

3. Experiment results and discussion

Three different experimental studies were carried out while developing and testing the SPAYK environment to demonstrate its capabilities. In the first experiment, the behavior of the Izhikevich neuron group was examined using data generated by regenerating the MNIST dataset samples as spiking signals. In addition, a single neuron is trained to respond to a specific label with synaptic plasticity (STDP). The second experiment replicated the results of the STDP study published by Masquelier et al. [23]. In the third experiment, unsupervised training for the classification problem was performed by adapted spiking MNIST dataset. Experiments and their results are presented in the following sections.

3.1. Izhikevich neuron group and supervised classifier

In this experiment, 3 random samples labeled 1, 4, and 7 from the MNIST dataset were converted into 500 ms long Poisson spike trains. To generate stimuli, these spike trains were arranged to repeat twice at random times. The resulting stimuli is given in Figure 6a. The MNIST dataset samples consist of 28×28 pixel images. These images were flattened and turned into a 784-length vector. The brightness values in this vector were mapped to the appropriate firing rates and a Poisson spike train with 784 neurons was generated. The firing rates in the stimuli varied over time to produce a realistic signal.

In the first part of the experiment, a group of 250 neurons was defined. About 20 percent of the neurons in the input stimuli were selected to form inhibitory synapses with neurons in the Izhikevich group. Each neuron in the Izhikevich group was randomly connected to 70 percent of the neurons in the stimuli. All synaptic connections have the same constant weight. The Izhikevich group was simulated with the input stimuli and the response was given in Figure 6b. As can be seen from the response plot, it was observed that the formed neuron group formed label-sensitive responses. This suggests that by improving the architecture, the neuron group can perform various tasks with synaptic plasticity.

In the second part of the experiment, a single Izhikevich neuron capable of STDP was defined. It is aimed to train this neuron to respond to a determined label by changing synaptic connection weights with plasticity. That whole training is intended to be performed under supervision. For this reason, the membrane potential signal of a regular spiking Izhikevich neuron that fires with a constant current only on the label 7 portions of the input stimuli was generated to train the neuron that can recognize label 7. This signal was determined as the supervision signal and is given in Figure 6c.

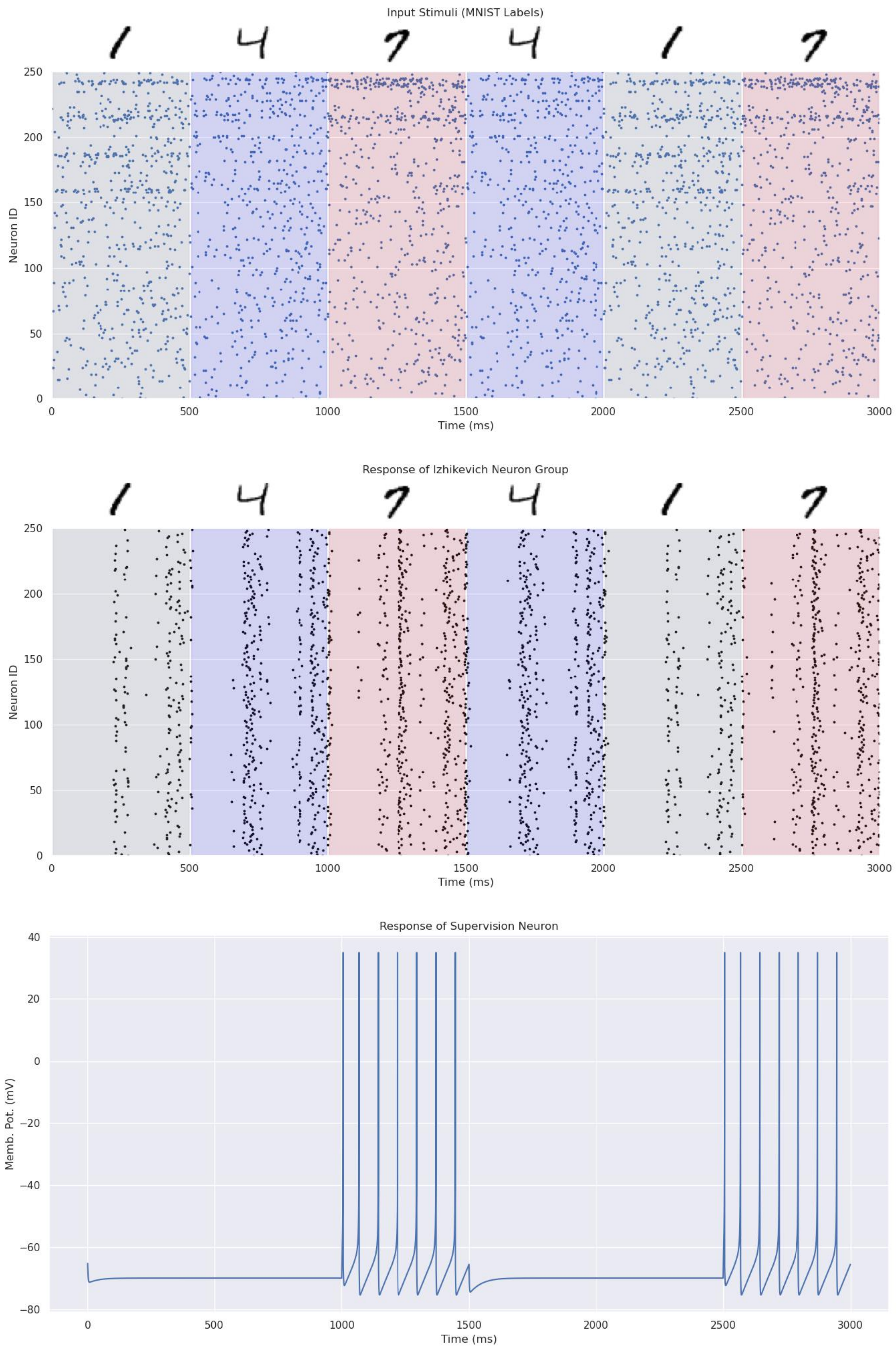

Figure 6. (a) The stimuli signal generated by repeating the spiking versions of the samples from 3 different labels, (b) the response of a group of 250 izhikevich neurons with synaptic connections to the input stimuli, (c) the supervision signal used to train a single neuron to recognize a specific label with STDP.

The STDP parameters for training were set to values that would give stable results by trial and error, based on the biologically consistent STDP parameters in Masquelier et al.'s paper [23]. A_+ and A_- were assigned values of 0.03125 and 0.025, respectively, and τ_+ and τ_- were assigned values of 16.8 and 30.7, respectively. Izhikevich neuron with STDP was simulated using stimuli and supervision signal and weights were calculated and recorded. Then the Izhikevich neuron using these weights is simulated. The graph of the membrane potential of the neuron over time is given in Figure 7a. As a result of the training, it was seen that synaptic plasticity can change the connections to give the desired result.

Time steps of weight-increasing and weight-decreasing LTP and LTD operations are displayed to analyze synaptic plasticity. The resulting graph, which is shown in Figure 7b, contains the LTP and LTD operations and also the IDs of the neurons in which they occur.

Figures can be reproduced with the script in the SPAYK repository at `experiments/exp_01_izhikevich_neurongroup_connections.py` and `experiments/exp_01_izhikevich_neuron_classifier_supervised.py`.

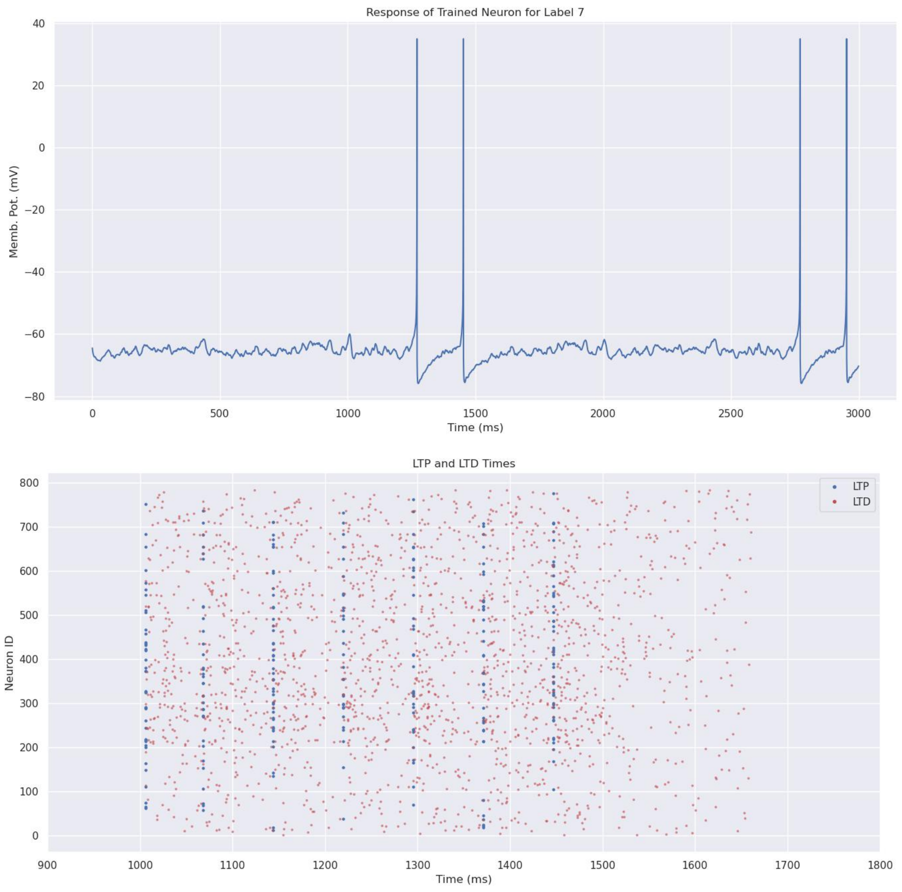

Figure 7. (a) The response of the simulated neuron using the weights produced as a result of the training, (b) an example section of LTP and LTD times.

3.2. STDP based pattern recognition

The experiment from Masquelier et al.'s study was performed to bring the spike response model simulation capability to the SPAYK environment [23]. It was demonstrated in the study that a neuron with a spike response model can detect repeating patterns in continuous spike trains by modifying weights via STDP. To replicate the study's findings, a single SRM LIF neuron with STDP capability was stimulated for 15 s with a special Poisson spike train of 2000 neurons. Parameters in the original study were used to produce the stimuli. The spike train was generated using neurons with firing rates ranging from 0 to 90 Hz. The firing rates of these neurons are changed at random at speeds ranging from +-360 Hz/s. Change speeds are modified at random between 1800 Hz/s and −1800 Hz/s. Furthermore, as in the original work, each neuron is guaranteed to fire at least once in 50 ms windows. A 50-ms segment of the resulting spike train was extracted and repeated at random intervals. As in the original study, the repeated part only includes the first 1000 neurons. Figure 8a shows a raster plot of a slice of the stimuli. Repeating patterns are highlighted with gray rectangles. The experiment demonstrated that a single SRM LIF neuron can detect repeating patterns with STDP. Figure 8b shows a graph cut of the membrane potential of the 15-s simulated neuron. Gray rectangles in the graph highlight repeating patterns. The graph shows how, over time, the neuron changes its weights to fire only within the patterns. Figures can be reproduced with the script in the SPAYK repository at `experiments/exp_02_srm_stdp_pattern_recog.py`.

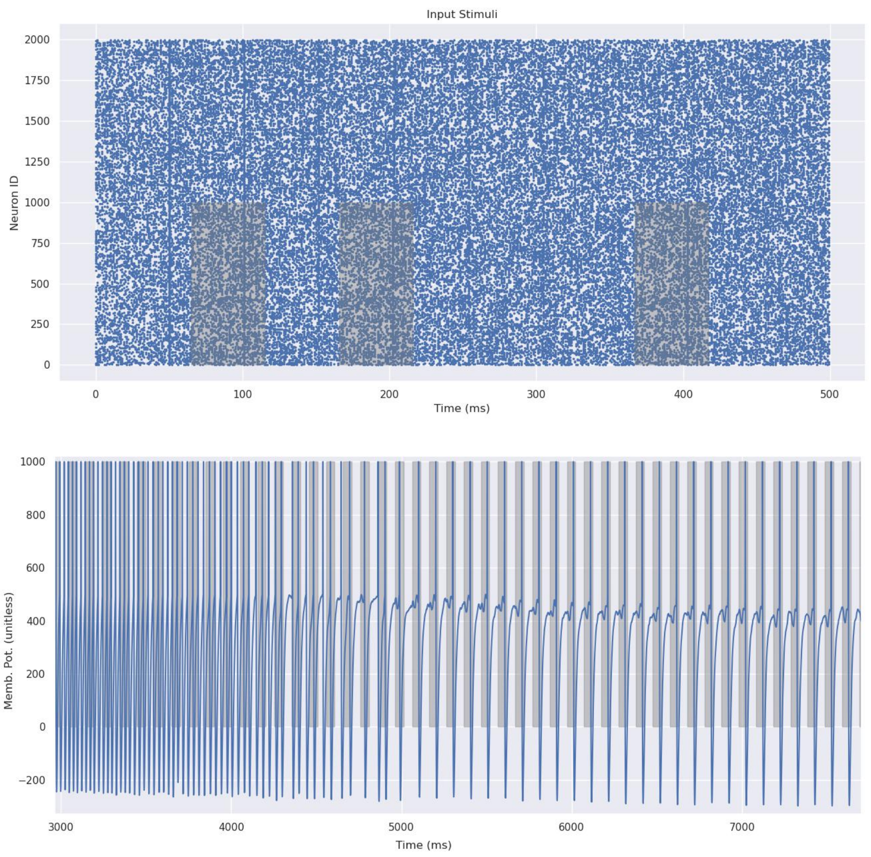

Figure 8. (a) Input Poisson spike train, (b) membrane potential of single SRM LIF neurons.

3.3. STDP based classification

In this experiment, the MNIST dataset was classified using a modified version of the unsupervised learning strategy from the prior experiment. The MNIST dataset is containing handwritten digits and is used in classification problems. The dataset consists of 28 × 28 pixels, grayscale coded images. These images were converted into firing patterns of 784 neurons, with firing rates directly proportional to the brightness values. Poisson spike train was obtained by using twice the number of neurons as in the previous experiment, and random 50 ms size spike patterns created from images with different labels were placed on the first 784 neurons. This created spike train was used as the base stimuli. The resulting stimuli is shown in Figure 9a.

A 50-ms pattern of a selected image is randomly spaced on this base spike train. This generated complex signal was used as stimuli. As a result of the experiment, a single SRM LIF neuron modified its weights in such a way that it fired in patterns with a size of 50 ms and a label of 5, repeating at random times in randomly arrayed patterns with different labels. Thus, in the presence of patterns of other labels, the classification of the determined label is achieved in an unsupervised manner. A section from the membrane potential of the neuron obtained as a result of the experiment is given in Figure 9b. In the figure, areas of the repeating pattern with the label 5 are highlighted with gray squares.

The values from the previous experiment were used as STDP parameters. A_+ and A_- were assigned values of 0.03125 and 0.025, respectively, and τ_+ and τ_- were assigned values of 16.8 and 33.7, respectively. Figures can be reproduced with the script in the SPAYK repository at `experiments/exp_03_srm_mnist_classifier_training.py`.

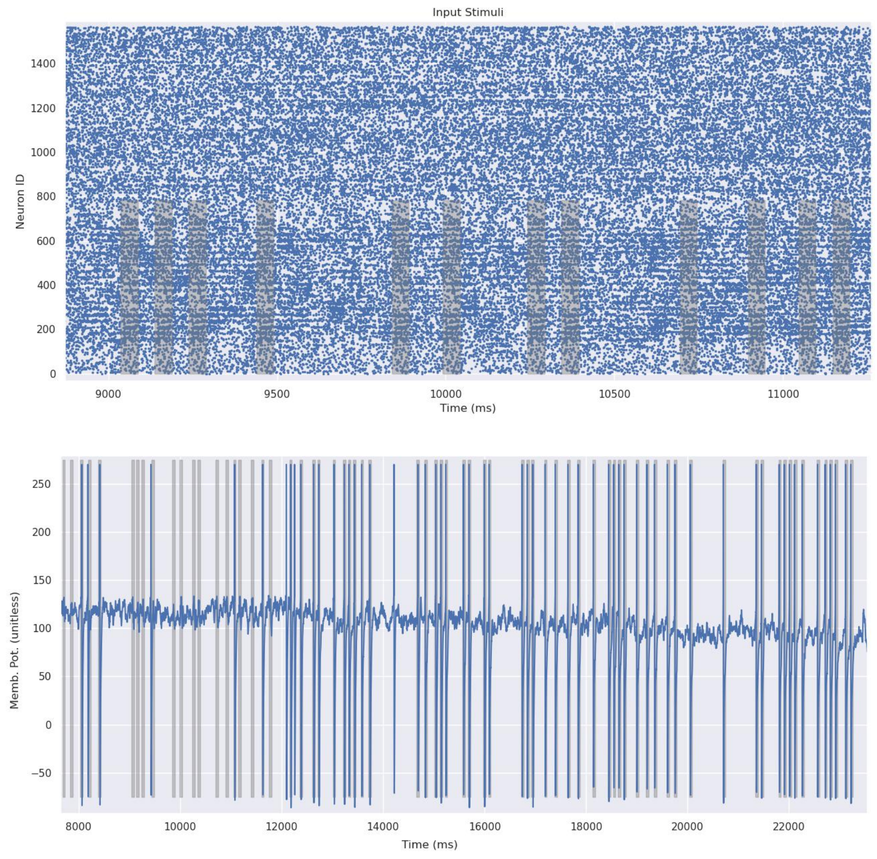

Figure 9. (a) Stimuli for MNIST classification, (b) membrane potential of SRM LIF neuron.

3.4. Performance analysis

Spayk's performance was evaluated by measuring the executation times and memory usage of groups with varying numbers of neurons and synapses. The measurement results are shown in Figure 10. The analysis was performed for three different groups, and these groups represent clusters of neurons that can be used in SPAYK experiments.

The first group consists of neurons presented in subsection 3.1. This group's neurons all have the

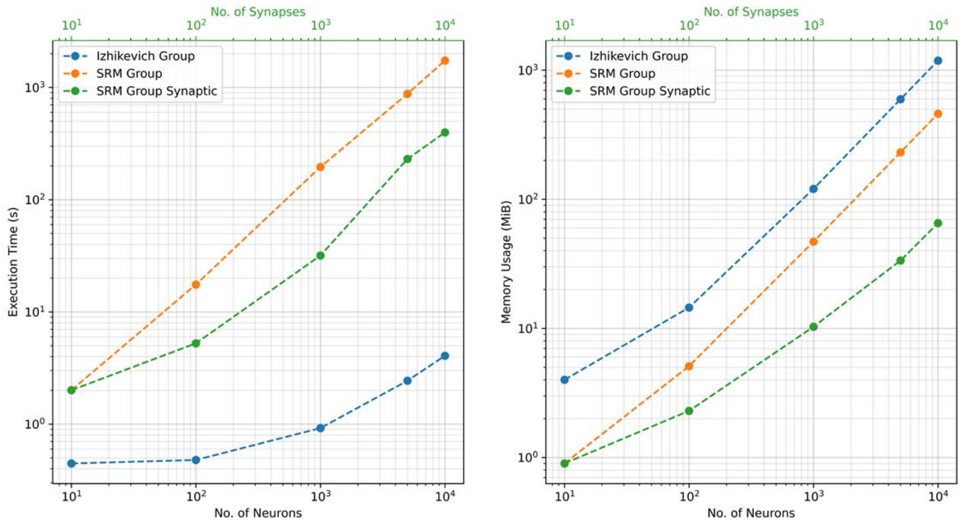

Figure 10. Execution time and memory usage of SPAYK simulations.

Izhikevich model and exhibit various random dynamics. The performance of this group was evaluated by varying the number of neurons in the group. The second group in the analysis consists of neurons with the SRM LIF model presented in other experiments. This group's performance was evaluated in two scenarios: the first in which the number of synapses remained constant while the number of neurons increased, and the second in which the number of neurons increased while the number of synapses remained constant. The horizontal axis at the bottom in Figure 10 indicates the number of neurons for the SRM Group, which is shown in orange, and the horizontal axis at the top indicates the number of the synapses for the SRM Group Synaptic, which is shown in green.

Because SPAYK's Izhikevich model implementation keeps the data of all neurons in large vectors and process them with large matrix operations, the runtime scales well as the number of neurons increases. Memory usage increases faster than the other model for the same reason. Because a main the loop iterating over all the neurons was used in the SRM LIF model's implementation for the group, the executation times are longer than in the Izhikevich model, but the amount of memory used is relatively low. Furthermore, increasing the number of synapses scales better than increasing the number of neurons. Experiments were run on a computer with an Intel(R) Core(TM) i7-3630QM processor running at 2.40GHz using 12 GB of RAM, running Ubuntu version 20.04.

4. Conclusion

Spiking neural network models run on neuromorphic hardware work very efficiently in terms of energy. For this reason, it is predicted that spiking neural networks will be a potential computational model for platforms such as robots with limited energy resources and continuous energy requirements. Neuromorphic hardware is still not widespread and it is very difficult to reach them today. For this reason, spiking neural network research is carried out in the form of simulations on traditional hardware such as CPU and GPU. There are a variety of simulation tools available for Spiking neural network designs. Most of these tools require a thorough understanding of the differential equations of neuron models and synaptic channels. This is a significant obstacle for researchers outside of neuroscience who want to use spiking networks to solve real-world engineering problems.

In this study, the SPAYK package, which is an environment for the simulation of spiking neural networks is presented. The main purpose of SPAYK is to create a framework that nonneuroscientists can easily use. SPAYK is released as an open-source framework. SPAYK has been developed to conduct practical experiments in spiking neural networks research and to simplify coding processes. In SPAYK, the Izhikevich model, spike response model and STDP are introduced. In the study, the experiments designed to demonstrate the features of SPAYK conducted, their source code and results are presented. The first of these experiments was to demonstrate the use of Izhikevich neuron groups and to examine the rhythms created by the neuron group. The experiment also aimed to demonstrate that a single Izhikevich neuron can recognize a particular pattern with STDP. It has been shown that an Izhikevich neuron can be trained to respond to a specific label by a supervision signal. Secondly, an experiment published in 2008 to recognize repetitive patterns with STDP was repeated and the results were verified. Finally, a classification problem adapted to spiking neural networks in the MNIST dataset is presented. These experiments are presented as sample applications to demonstrate the capabilities of the SPAYK environment. In this study, it has been shown that synaptic plasticity can be used for training a network in a supervised or unsupervised manner. This result demonstrates that tasks that require a large number of neurons in the artificial neural network can be performed by a single or a small number of spiking neurons. The results suggest that performing the calculations with the spiking networks would be more efficient. In addition, it is thought that the potential of such networks will be unleashed with the rapidly developed neuromorphic platforms.

Spiking neural network simulators have different limitations according to their intended use. Total simulation time is very important in studies carried out to investigate brain regions or to examine group behavior in a large number of nerve cells. To reduce the total simulation time, simulators offer support for parallelizing processes on GPU hardware. Since SPAYK is designed to study real-time signal processing or robotics problems, it must have the same hardware acceleration support to offer real-time work. Until neuromorphic platforms become widespread, hardware acceleration support becomes mandatory as the networks designed to be forced will run on CPUs and GPUs. Work continues for SPAYK's Numba just-in-time compiler (JIT) support. This is the most important current limitation of SPAYK.

SPAYK is a project that is open to continuous development and has great potential. As SPAYK will be continuously developed, changes will occur in the documentation and class structures. SPAYK is planned to be developed in the form of major and minor versions. Providing neuromorphic hardware support, network visualization and a visual user interface are determined as future research directions. SPAYK was developed to facilitate the spiking neural network experiments. This makes the required code and architecture more understandable than other simulators and makes SPAYK a potential tool for facilitating experimental robotics and computer vision research based on spiking neural networks. It is thought that the great potential of various bioinspired sensors such as event-based cameras will deeply affect these areas with spiking neural networks.

References

[1] Maass W. Networks of spiking neurons: The third generation of neural network models. Neural Networks 1997; 10 (9): 1659–1671. doi:10.1016/s0893-6080(97)00011-7

[2] Cao Z, Cheng L, Zhou C, Gu N, Wang X et al. Spiking neural network-based target tracking control for autonomous mobile robots. Neural Computing and Applications 2015; 26 (8): 1839–1847. doi:10.1007/s00521-015-1848-5

[3] Cheng X, Hao Y, Xu J, Xu B. LISNN: Improving spiking neural networks with lateral interactions for robust object recognition. In: Twenty-Ninth International Joint Conference on Artificial Intelligence; Yokohama, Japan; 2020.

doi:10.24963/ijcai.2020/211

[4] Kim S, Park S, Na B, Yoon S. Spiking-YOLO: Spiking neural network for energy-efficient object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence; New York, USA; 2020. pp. 11270–11277. doi:10.1609/aaai. v34i07.6787

[5] Wu J, Chua Y, Zhang M, Li H, Tan KC. A spiking neural network framework for robust sound classification. Frontiers in Neuroscience 2018;12 doi:10.3389/fnins.2018.00836

[6] Yan Z, Zhou J, Wong WF. Energy efficient ECG classification with spiking neural network. Biomedical Signal Processing and Control 2021; 63: 102170. doi:10.1016/j.bspc.2020.102170

[7] Tikidji-Hamburyan RA, Narayana V, Bozkus Z, El-Ghazawi TA. Software for brain network simulations: A comparative study. Frontiers in Neuroinformatics 2017;11 doi:10.3389/fninf.2017.00046

[8] Goodman D. Brian: a simulator for spiking neural networks in python. Frontiers in Neuroinformatics 2 2008; doi:10.3389/neuro.11.005.2008

[9] Gewaltig MO, Diesmann M. Nest (neural simulation tool). Scholarpedia 2007;2 (4) :1430

[10] Hines ML, Carnevale NT. Neuron: A tool for neuroscientists. The Neuroscientist 2001; (7): 123–135. doi:10.1177/107385840100700207

[11] Wilson MA, Bhalla US, Uhley JD, Bower JM. GENESIS: A System for Simulating Neural Networks. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA: 1989, pp. 485–492.

[12] Chou TS, Kashyap HJ, Xing J, Listopad S, Rounds EL et al . CARLsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters. In: International Joint Conference on Neural Networks (IJCNN) 2018; doi:10.1109/ijcnn.2018.8489326

[13] Eliasmith C, Anderson CH. Neural Engineering (Computational Neuroscience Series): Computational Representation and Dynamics in Neurobiological Systems. Cambridge, MA, USA: MIT Press, 2002.

[14] Bekolay T, Bergstra J, Hunsberger E, DeWolf T, Stewart TC et al. Nengo: a python tool for building large-scale functional brain models. Frontiers in Neuroinformatics 2014;7 doi:10.3389/fninf.2013.00048

[15] Vitay J, Dinkelbach HU, Hamker FH. ANNarchy: a code generation approach to neural simulations on parallel hardware. Frontiers in Neuroinformatics 2015;9 doi:10.3389/fninf.2015.00019

[16] Mozafari M, Ganjtabesh M, Nowzari-Dalini A, Masquelier T. SpykeTorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. Frontiers in Neuroscience 13 2019; doi:10.3389/fnins.2019.00625

[17] Fang W, Chen Y, Ding J, Chen D, Yu Z et al. Spikingjelly. https://github.com/fangwei123456/spikingjelly, accessed: 2022-11-01.

[18] Lee JH, Delbruck T, Pfeiffer M. Training deep spiking neural networks using backpropagation. Frontiers in Neuroscience 10 2016; doi:10.3389/fnins.2016.00508.18

[19] Neftci EO, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine 2019; 36 (6): 51–63. doi:10.1109/msp.2019.2931595

[20] Wang X, Lin X, Dang X. Supervised learning in spiking neural networks: A review of algorithms and evaluations. Neural Networks 2020;125 :258–280. doi:10.1016/j.neunet.2020.02.011

[21] Li X, Yi H, Luo S. Pattern recognition of spiking neural networks based on visual mechanism and supervised synaptic learning. Neural Plasticity 2020; 1–11. doi:10.1155/2020/8851351

[22] Liu J, Huo H, Hu W, Fang T. Brain-inspired hierarchical spiking neural network using unsupervised STDP rule for image classification. In: 10th International Conference on Machine Learning and Computing 2018; doi:10.1145/3195106.3195115

[23] Masquelier T, Guyonneau R, Thorpe SJ. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. PLoS ONE 2018;3 (1): e1377. doi:10.1371/journal.pone.0001377

[24] Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology 1952;117 (4): 500–544. doi:10.1113/jphysiol.1952.sp004764

[25] Skocik MJ, Long LN. On the capabilities and computational costs of neuron models. IEEE Transactions on Neural Networks and Learning Systems 2015; 25 (8): 1474–1483. doi:10.1109/tnnls.2013.2294016

[26] Izhikevich E. Simple model of spiking neurons. IEEE Transactions on Neural Networks 2003;14 (6): 1569–1572. doi:10.1109/tnn.2003.820440

[27] Gerstner W, Kistler WM, Naud R, Paninski L. Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge: Cambridge University Press, 2014. doi:10.1017/CBO9781107447615

[28] Gerstner W. Spike-response model. Scholarpedia 2018;3 (12): 1343. doi:10.4249/scholarpedia.1343

[29] Hebb DO. The organization of behavior: A neuropsychological theory. Brain Res Bull, New York, USA: Wiley, 1949.