Int. J. Appl. Math. Comput. Sci., 2024, Vol. 34, No. 3, 467–483
DOI: 10.61822/amcs-2024-0033

A SPIKING NEURAL NETWORK BASED ON THALAMO–CORTICAL NEURONS FOR SELF–LEARNING AGENT APPLICATIONS

DAMIAN HUDEREK [a], SZYMON SZCZĘSNY [a,*], PAWEŁ PIETRZAK [a], RAUL RATO [b], ŁUKASZ PRZYBOROWSKI [a]

[a]Faculty of Computing and Telecommunications
Poznań University of Technology
Piotrowo 3A, 61-138 Poznań, Poland
e-mail: szymon.szczesny@put.poznan.pl

[b]Department of Electrical and Computer Engineering
NOVA University of Lisbon
Quinta da Torre, 2829-516 Caparica, Portugal

The paper proposes a non-iterative training algorithm for a power efficient SNN classifier for applications in self-learning systems. The approach uses mechanisms of preprocessing of signals from sensory neurons typical of a thalamus in a diencephalon. The algorithm concept is based on a cusp catastrophe model and on training by routing. The algorithm guarantees a zero dispersion of connection weight values across the entire network, which is particularly important in the case of hardware implementation based on programmable logic devices. Due to non-iterative mechanisms inspired by training methods for associative memories, the approach makes it possible to estimate the capacity of the network and required hardware resources. The trained network shows resistance to the phenomenon of catastrophic forgetting. Low complexity of the algorithm makes in-situ hardware training possible without using power-hungry accelerators. The paper compares the complexities of hardware implementations of the algorithm with the classic STDP and conversion procedures. The basic application of the algorithm is an autonomous agent equipped with a vision system and based on a classic FPGA device.

Keywords: self-learning systems, classification, spiking neural networks, feature recognition, learning by routing.

1. Introduction

In addition to cloud solutions, where various artificial intelligence (AI) methods are often used, edge computing is becoming an increasingly common approach in the IoT (Internet of things). Research proves that AI-based edge accelerators make it possible to achieve comparable and in many cases better performance in terms of power or cost than traditional cloud servers (Liang *et al.*, 2020). In edge applications, various computer vision techniques are widely used, especially neural networks (Raha *et al.*, 2021). However, nowadays, the development of neural networks in this type of applications faces serious limitations. Providing high precision deep networks, implemented using the so-called 2nd generation neurons, is difficult in mobile devices for several reasons. First of all, deep networks are characterized by a large number of parameters (e.g., Resnet152 (He *et al.*, 2015) or VGG (Leong *et al.*, 2020)). The number of parameters in this type of networks is tens of thousands of times greater than that of DSP (digital signal processor) blocks dedicated to their implementation in most FPGA (field programmable gate array) edge devices. Secondly, applications in mobile devices often require real-time processing, which is especially difficult to obtain using limited resources in the hardware layer. The hardware implementation of 2nd generation networks in edge devices often requires using dedicated IPCores CMOS (Szczęsny, 2017). Designing them is a time-consuming task and prototyping is very expensive. The alternative is 3rd generation networks, i.e., spiking neural networks (SNNs). They are used, among others, in image analysis (Meftah *et al.*, 2010), pattern classification (Nazari *et al.*, 2020) and sound processing

[*]Corresponding author

(Encke and Hemmert, 2018).

There are high hopes concerning spiking networks due to better perception properties compared with 2nd generation networks. This is confirmed by examples of solving nonlinear problems using a single neuron (Rowcliffe *et al.*, 2006) or the possibility of real-time processing of biological signals using only two-layer SNNs (Szczęsny *et al.*, 2021). They are also employed in medical sciences for broadly understood modeling of interactions in the nervous system (Bartłomiejczyk *et al.*, 2023). For all of the above reasons, research centers an present biology-inspired neuroprocessors (Pisarev *et al.*, 2020; Mayr *et al.*, 2019), and IT companies are increasingly investing in their own neuroprocessors: IBM—the True North processor (Cheng *et al.*, 2017) or Intel—the Loihi processor (Davies *et al.*, 2018), which already has a 2nd generation. There are other popular architectures that are used, e.g., in image processing or automotive industry: BrainScaleS, Neurogrid, SpiNNaker and Akida.

Using dedicated neuromorphic hardware allows reduction in power consumption of final implementations. SNNs are several to several dozen times more energy efficient compared with the currently popular CNNs (convolutional neural networks) (Wu *et al.*, 2022). Despite this, new methods are still being sought to improve the energy efficiency of these networks, e.g., through choosing the right architectures (Na *et al.*, 2022), learning with simultaneous power optimization (Neftci *et al.*, 2019) or tuning the learnable membrane time constant (Fang *et al.*, 2021). Additionally, a constant challenge in the context of energy efficiency remains the training of SNNs. Existing methods are highly iterative in nature and some are very hyper-parameter-sensitive. SNN iterative learning algorithms are also expensive due to the time dependence between the signals in the network. The learning time with iterative methods is long due to the need to perform timed simulations at each step of the learning algorithm. This is costly with standard computer hardware even with parallel computing. The heavily reduced computational resources of edge devices are not dedicated to implementing such simulations. In addition, the activation function of a spiking neuron is nondifferentiable, which further increases the cost of learning with, for example, backpropagation (SLAYER) algorithms (Shrestha and Orchard, 2018). For these reasons, training can be very complex. The most important methods include (Li *et al.*, 2020) gradient descent methods, reinforcement learning methods, STDP (spike-timing-dependent plasticity) methods and ensemble learning methods.

Gradient descent cannot be used for SNNs out of the box due to the nondifferentiable nature of such models. However, multiple adaptation approaches have been developed over the years (Li *et al.*, 2020; Wu *et al.*, 2018b). These methods can yield comparable results to deep neural networks but are computationally heavy and limiting in terms of biologically plausible network architectures. Reinforcement learning is also investigated in terms of SNNs. For example, in the work of Ponghiran *et al.* (2019) small spiking networks were trained with success using Q-learning to play Atari games.

The existing approaches in this field are highly iterative in nature. Of particular interest is the STDP algorithm. It describes a mechanism for strengthening/weakening synapses based on presynaptic and postsynaptic spike timings. It falls under the category of unsupervised learning algorithms, and there is evidence that a similar mechanism is used in the brain (Markram *et al.*, 2012; Tazerart *et al.*, 2019). Sometimes some supervised learning algorithms are partially based on the STDP mechanisms (Ponulak, 2008). However, the STDP algorithm is not well developed yet and is difficult to use for solving practical machine learning problems.

Most of all tuning of hyper-parameters can be difficult and, if done incorrectly, it may render training impossible. Apart from that, the need to compute layers' responses in individual time steps makes the nature of the algorithm to be highly iterative. Due to this fact, usage of GPUs (graphics processing units) does not give the same speed-up as in gradient descent methods for second generation neural networks. Lastly, ensemble learning methods also exist for spiking neural networks (Neculae, 2020; Hazan *et al.*, 2018; Kozdon and Bentley, 2017). These methods usually focus on training multiple models in parallel and aggregating a unified response from them. Depending on the hardware resources, these can provide some performance gains due to distributed processing (Kozdon and Bentley, 2017), but they still use regular learning algorithms such as STDP to train single models and thus suffer from the same shortcomings.

A separate problem is preparing a comprehensive training set. For SNNs there are two choices: standard machine learning datasets and neuromorphic datasets. A great example of a standard machine learning dataset is ImageNet (Deng *et al.*, 2009)—a dataset used for computer vision competition called ImageNet large scale visual recognition challenge (ILSVRC). In the field of deep learning, state-of-the-art-algorithms were presented each year at this competition. Neuromorphic datasets on the other hand, differ greatly from standard ones as they are prepared in a way to mimic the stimuli acquisition in the human brain. For computer vision, these datasets are prepared using neuromorphic cameras and reside in the form of event streams with added noise (Li *et al.*, 2017; Wu *et al.*, 2018a). Examples of these include N-MNIST and DVS-CIFAR-10. Neuromorphic datasets may be a key to progressing the field of spiking neural networks, but, unfortunately, they are very hard to prepare as the preparation is largely manual and involves scanning images with image sensors to generate events (Li *et al.*, 2017; Wu *et al.*, 2018a). Because of that, there are not many neuromorphic datasets publicly available.

Despite the complexity of training algorithms, spiking networks are increasingly popular. As mentioned, one of the areas of their usage is autonomous vehicles. Spiking neural networks are used in the analysis of data from dynamic vision sensors (Xuelei, 2023), LIDAR sensors (Albert *et al.*, 2021) or for energy-efficient control (Raz *et al.*, 2023). The literature describes applications of SNN networks in a control task of unmanned aerial vehicles (UAVs) (Stagsted *et al.*, 2020), computation and control in space (Pereira-Pires *et al.*, 2019), and in examples of obstacle detection using neuroprocessors (Salt *et al.*, 2020) or solutions based on the Loihi processor for applications in the automotive industry (Viale *et al.*, 2021). On the other hand, most modern concepts of using artificial intelligence in mobile vehicles assume self-supervised learning (Cech *et al.*, 2021; Vosahlik *et al.*, 2021). However, such solutions are mainly based on 2nd generation deep networks, which are expensive to implement. An additional problem is managing training data (Zhao *et al.*, 2021). Taking into account the popularity of edge computing, the effectiveness of SNNs, their growing popularity in mobile vehicles and limitations of current autonomous systems based on artificial intelligence, we propose an effective SNN training algorithm dedicated to image processing in edge devices.

Due to the potential application of SNNs in image processing, the solution described in the paper is based on the functionality of a thalamus, which is a part of a diencephalon. The thalamus is responsible for the initial evaluation of signals coming from all senses, except for smell, and sending them to the cerebral cortex (Torrico and Munakomi, 2020). One of the basic functionalities of the thalamus is preprocessing signals from retinal axons of ganglion cells (Oster *et al.*, 2004). Inhibition phenomena observed in the thalamus allow thalamocortical operations to dynamically match ongoing behavioral demands (Halassa and Acsády, 2016). One of the basic anatomical features of the thalamus is the presence of atypical inhibition-induced spiking neurons (Hua *et al.*, 2022), whose model and possibility of application in the classification task are described in more detail in this paper. Due to the potential use in self-learning agent applications and usually highly limited resources in the hardware layer of edge devices, we define several key assumptions for the network routing algorithm:

- *pruning:* limiting network complexity by reducing the number of synaptic connections and decreasing the dispersion of synaptic weight values to reduce the usage of hardware resources during implementation;
- *quantization:* increasing network capacity to provide more memory in the network, the size of which is strongly limited by hardware resources;
- *simplification:* providing a simple noniterative training algorithm for the network, with the possibility to find synaptic connections. This approach makes it possible to train new patterns in an adaptive mobile system, as long as the complexity of the training algorithm is low.

The paper is organized as follows. Section 2 describes models of neurons of the thalamus used in the research. Section 3 presents an architecture of an SNN based on the anatomy of the thalamus. Section 4 focuses on the network training algorithm using single patterns and including the reduction in weight dispersion. Section 5 presents the results of network operation tests concerning the problem of classifying everyday objects and compares the hardware complexity of the algorithm with other iterative SNN learning algorithms. The paper ends with the summary of research results and a discussion on the areas of their application.

2. T-C and C-C model

The architecture of a neural network dedicated to image processing is based on the activity of a diencephalon responsible for quick processing of a large number of signals (Lim and Golden, 2007). These include ones related to the following functions: sensory (touch, pain, temperature), taste, hearing, attention, motor skills, emotions, partly hippocampal functions and, above all, the visual functions. Section 2.1 describes models of neurons which build the thalamus, which makes up most of the gray matter in the diencephalon. Section 2.2 describes an effective model of a neuron based on the cusp catastrophe, used, for example, for modeling emotions and designing neuro-fuzzy decision systems.

2.1. Thalamo-cortical (T-C) neuron model. In this research we use the Izhikevich model (Izhikevich, 2004), which, among many models of spiking neurons, is characterized by low complexity and, at the same time, high accuracy of mapping biological aspects of real-life neurons (Abusnaina and Abdullah, 2014). This model is described by Eqns. (1), (2), (3) consisting of membrane potential u and membrane recovery v variables. I_{app} is the sum of synaptic currents flowing into the soma of the neuron:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = 0.04v^2 + 5v + 140 - u + I_{\text{app}}, \tag{1}$$

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

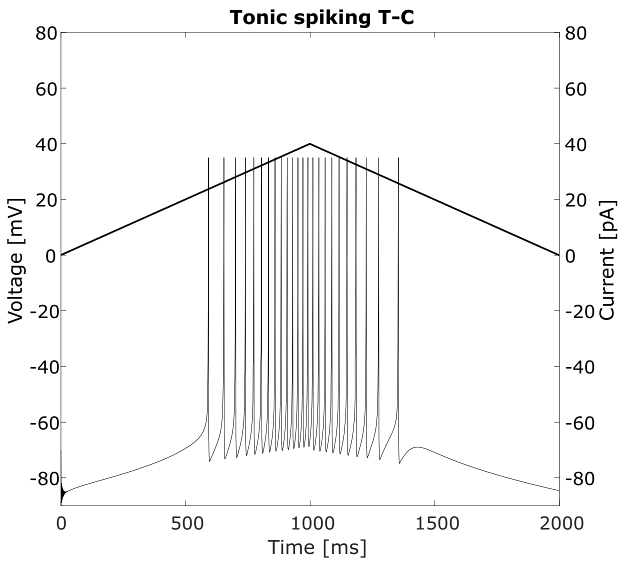

Fig. 1. Response of a tonic spiking neuron (TS/T-C) to a triangular excitation.

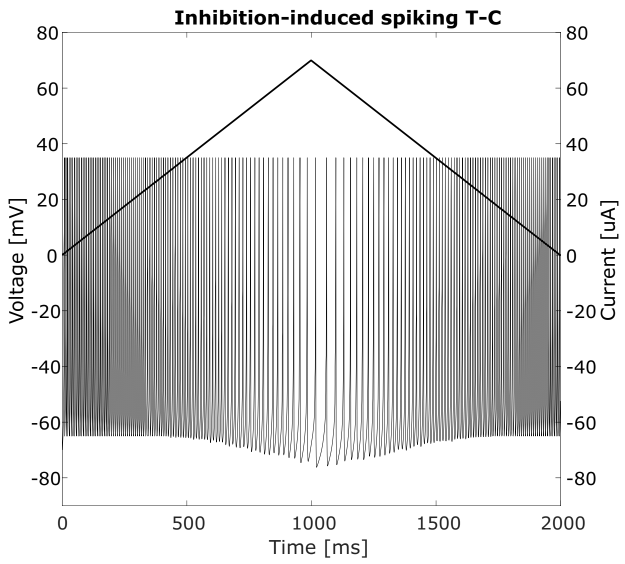

Fig. 2. Response of an inhibition-induced spiking (IIS/T-C) neuron to a triangular excitation.

$$\text{if} \quad v \geq 30\,\text{mV} \quad \text{then} \quad \begin{cases} v \leftarrow c, \\ u \leftarrow u + d. \end{cases} \tag{3}$$

Dimensionless parameters a, b, c, d make it possible to easily model one of the twenty types of neurons defined by Izhikevich (2004). In the task of modeling the activity of the thalamus, we used two models in particular. The first one is the common tonic spiking model obtained for the following set of parameters: $[a, b, c, d] = [0.02, 0.2, -65, 6]$. Its response to the triangular excitation is shown in Fig. 1.

The second model used is of the inhibition-induced spiking type. As Izhikevich noted, a strange feature of many neurons in the thalamus region is that they spike when hyperpolarized by an inhibitory input (Izhikevich, 2004). The justification for this phenomenon is deactivation of the calcium current, which leads to tonic spiking during the injection of current into a soma. The inhibition-induced spiking model is obtained for the following set of parameters: $[a, b, c, d] = [-0.02, -1, -65, 8]$. Its response to the triangular excitation is shown in Fig. 2. Notice the decrease in the frequency of spikes along with the increase in the soma current, and thus the complementary nature of operation of inhibition-induced neurons in relation to tonic spiking neurons. It can be also noticed that the activation of inhibition-induced spiking neurons requires the application of almost twice as high currents to the soma. Moreover, the frequency of generated spikes in stimulated inhibition-induced neurons is higher than that of spikes generated by tonic spiking neurons. Both the larger currents flowing into the soma, the higher frequency of spikes force the limitation of the number of inhibition-induced neurons, e.g., by using them only in one layer.

2.2. Cusp-catastrophe (C-C) neuron model.

The second type of neurons used in our research is a model based on the cusp catastrophe (Chen *et al.*, 2016), which uses probability density functions to model sudden changes. This model is used, among others, in social behavior analysis and explanation of health outcomes (Chen *et al.*, 2014), behavioral pathology during the use of psychoactive and addictive substances (Guastello *et al.*, 2008) and HIV prevention (Chen *et al.*, 2013). Currently, it is employed used in designing deep neural networks (Daw and He, 2020) as one of the simplest models of a catastrophe. In this paper we decide to use the cusp catastrophe model due to the use of the pattern mapping approach employing a sequence of synaptic connections in the network and ignoring the influence of connection weights. The sensitivity of the cusp catastrophe model to sudden changes of input is used to model the propagation of a nerve spike induced by the detection of features characteristic for individual classes of patterns.

The practical application of the cusp catastrophe model is discussed using the example of the classifier problem. Figure 3(a) shows an example of a pattern, whose features are decomposed in Fig. 3(c). Figure 3(b) shows a noisy pattern, and the decomposition of its features is shown in Fig. 3(d). The probability P_{pattern} of recognizing the noisy pattern depends on the sum of probabilities P_ϕ of recognizing individual features ϕ in

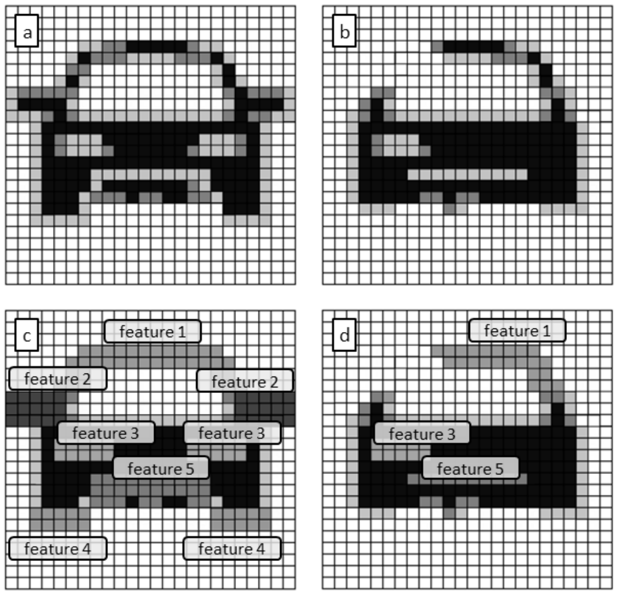

Fig. 3. Problem of classification based on the evaluation of features: original pattern (a), noisy pattern (b), decomposition of features of the original pattern (c), decomposition of features of the noisy pattern (d).

accordance with

$$P_{\text{pattern}} \sim \frac{\sum P_\phi}{\sum \phi}, \tag{4}$$

where the parameter ϕ is the index numbering the features. The classifier operation defined as such, however, leads to the need to provide the neural network with mechanisms for estimating the probability of detecting features. Such an approach also leads to a relatively large dispersion of weights in the network, which are the basis of the aforementioned estimation mechanism. Large dispersions of weight values are disadvantageous due to the cost of hardware implementation. As the dispersion increases, the required amount of hardware resources needed to represent network weights increases.

For example, weight implementation using an FPGA technology requires the use of dedicated DSP blocks, while zero-dispersion implementation does not. Implementation of weights with a dispersion of 100 and with no DSP blocks requires about three times as many LUTs (lookup tables) as the zero-dispersion implementation. Meanwhile, the significance of a given feature in defining the belonging of a pattern to a class is not equivalent to the weight of this feature in the analyzed image, i.e., the amount of data that represents this feature. An example of an axiom is having the *wheels* feature by the *car* pattern. In this case Feature 4 (wheels), characteristic for the pattern in Fig. 3, is of greater significance than, for example, Feature 1 (roof), but the weight, i.e., the number of pixels in the pattern, and thus the number of synaptic connections in the classifier, is exactly the opposite. The pattern in Fig. 3(d) cannot be classified as *car*. The reason, however, is not the lack of Feature 2 or the low weight of Features 1, 3 and 5, but the lack of Feature 4. Such situations lead to a specific non-equilibrium of the significance of features—some features are preponderant in the classification process. For this reason, Eqn. (4) cannot hold. The problem of significance of features is one of the basic challenges in creating classifiers. Its solutions are sometimes very expensive to implement due to the need to grow the network model (Chu *et al.*, 2019) or to provide mechanisms for removing feature interaction (Rajbahadur *et al.*, 2022). In order to effectively detect features by the SNN implemented using edge solutions, it is necessary to propose a simple feature weighing algorithm. The result of the evaluation of features should constitute the basic criterion, both in the process of pattern classification by the trained network and in the earlier process of training the network, as the basis for recognizing the pattern presented at the network input as the new one, i.e., that which can be remembered in the network.

In spiking neural networks, the membrane potential of a neuron depends on both the activity of the neurons in the previous layer, to which it is connected, and its current state, e.g., the temporary phase of refraction. Therefore, it is more efficient to sum the currents flowing into the neuron, which depend less on its current state (whether it is firing or not) and more on the activity of neurons in the previous layer. In a spiking network stimulated by current signals, the probability of detecting a given feature ϕ is proportional to currents $i(\phi)$ representing the given feature in the pattern according to

$$P_\phi \sim \sum i(\phi). \tag{5}$$

Without an additional mechanism of weighing features, the claim about their different significance is not fulfilled according to (5). Please bear in mind that electromagnetic variables like currents or tensions can be either positive or negative and that probabilities are always positive while probability amplitudes can be positive, negative or complex. The use of probability amplitudes for describing an SNN is beyond the scope of this work and will be addressed in a future paper.

For each feature ϕ it is possible to additionally define current threshold i_{th}, exceeding of which means an unambiguous identification of the feature. The binary understanding of the probability of detecting particular features in such a model is defined by

$$P_\phi(i_{\text{th}}) = \begin{cases} 1 & \text{if } \sum_{i(\phi)} \geq i_{\text{th}}, \\ 0 & \text{if } \sum_{i(\phi)} < i_{\text{th}}. \end{cases} \tag{6}$$

The feature ϕ is recognized or not depending on the value of the sum of all currents generated in the network for the given feature presented at the network input.

However, the adoption of such a simple classification method is impossible, primarily due to the high sensitivity of the classifier's response, especially in the vicinity of the threshold i_{th}. A separate problem would be the selection of training data to establish the threshold parameter, which conflicts with the self-learning agent concept. Focusing on the implementation of biological learning mechanisms, we used the cusp catastrophe model (CCM) to identify features.

The mathematical description of the cusp catastrophe defines a model of a dynamic system according to (Chen *et al.*, 2016)

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{\mathrm{d}V(z;\alpha,\beta)}{\mathrm{d}z}, \tag{7}$$

where

$$V(z;\alpha,\beta) = \alpha z + \frac{1}{2}\beta z^2 - \frac{1}{4}z^4. \tag{8}$$

The model includes control factors α and β, which determine output variable z. In the case of spiking neurons, the parameter controlling the behavior of a neuron is the total synaptic current flowing into the soma, while the output variable is the frequency of changes in the functional potential of the axon, i.e., the generation of voltage impulses. For simplicity, we assume that the model does not use the bifurcation parameter, i.e., $\beta = 0$. The so-defined model of a spiking neuron based on the CCM is described by Huderek *et al.* (2019). Figure 4 features characteristics of C-C spiking neurons based on the CCM model for both types of T-C neurons used to implement the thalamus model: tonic spiking (TS) and inhibition-induced spiking (IIS).

In addition to the dual nature of both types of T-C neurons (TS and IIS), attention is drawn to the hysteresis between the values of currents i_{th1} and i_{th2}. Due to its presence, equations for the frequency of spikes f are met depending on the sum of all synaptic currents $\sum_i$ that flow into the selected neuron: Eqn. (9) in the case of the TS/C-C neuron, i.e., of the tonic spiking type (Huderek *et al.*, 2019) and Eqn. (10) in the case of the IIS/C-C neuron, i.e., of the inhibition-induced spiking type,

$$f_{\text{TS/CC}}(t) = \begin{cases} f_C & \text{if } \sum_i > i_{\text{th2}}, \\ f_C & \text{if } \sum_i > i_{\text{th1}} \wedge f(t-1) = f_C, \\ f_A & \text{if } \sum_i < i_{\text{th2}} \wedge f(t-1) = f_A, \\ f_A & \text{if } \sum_i < i_{\text{th1}}, \end{cases} \tag{9}$$

$$f_{\text{IIS/CC}}(t) = \begin{cases} f_A & \text{if } \sum_i < i_{\text{th1}}, \\ f_A & \text{if } \sum_i < i_{\text{th2}} \wedge f(t-1) = f_A, \\ f_C & \text{if } \sum_i > i_{\text{th1}} \wedge f(t-1) = f_C, \\ f_C & \text{if } \sum_i > i_{\text{th2}}. \end{cases} \tag{10}$$

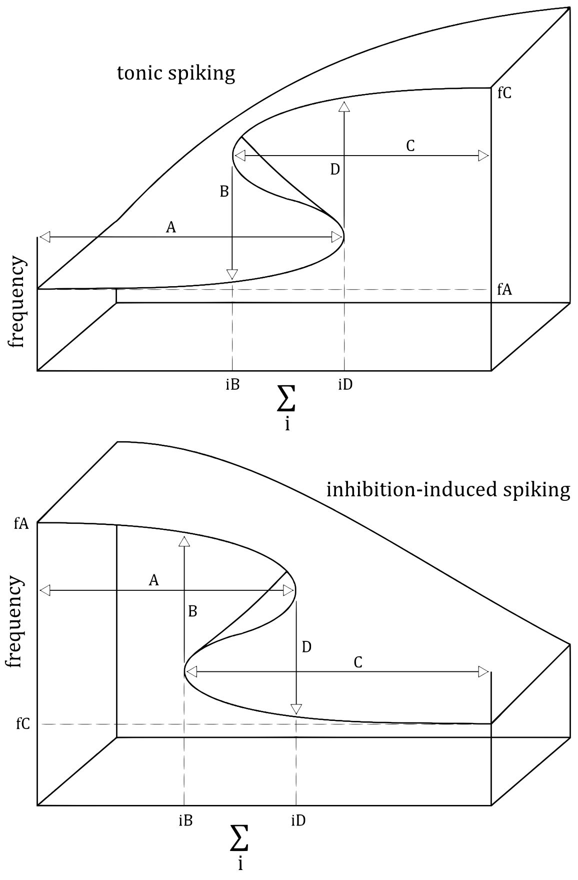

Fig. 4. Cusp catastrophe model for both types of neurons: tonic spiking and inhibition-induced spiking (Huderek *et al.*, 2019).

The original neuron model defined by Eqns. (1)–(3) was modified with the cusp catastrophe model described by Eqns. (10) and (9). After this modification, time responses of T-C neurons based on a C-C model for a triangular synaptic current are presented in Fig. 5 for the Tonic Spiking (TS/C-C) type and in Fig. 6 for the Inhibition-Induced (IIS/C-C) type.

Using the cusp catastrophe model described above to estimate the probability of pattern classification based on features leads to a three-state model. Due to the hysteresis of the C-C model, Eqn. (6), when using the model, takes the following form:

$$P_\phi(i_{\text{th}}) = \begin{cases} 1 & \text{if } \sum_{i(\phi)} \geq i_{\text{th2}}, \\ \text{is unknown} & \text{if } i_{\text{th1}} \leq \sum_{i(\phi)} < i_{\text{th2}}, \\ 0 & \text{if } \sum_{i(\phi)} < i_{\text{th1}}. \end{cases} \tag{11}$$

This approach offers a large margin of error and increases the probability of a correct classifier response. It should be noted that the large margin of error is the main advantage of using the CCM and the C-C neurons defined

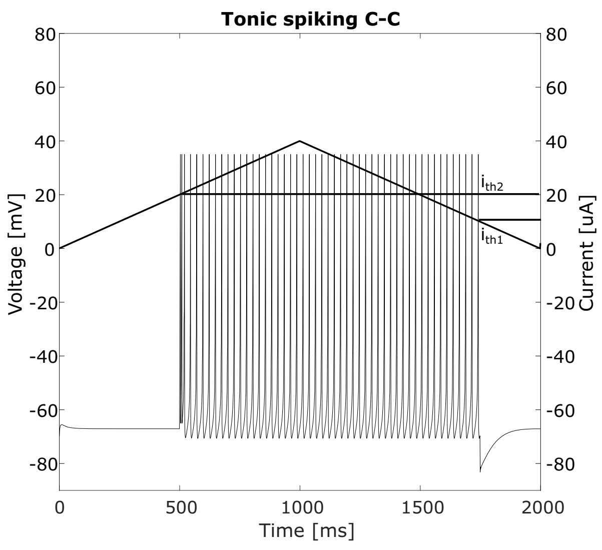

Fig. 5. Response of a tonic spiking neuron (TS/C-C) based on a C-C model to a triangular excitation.

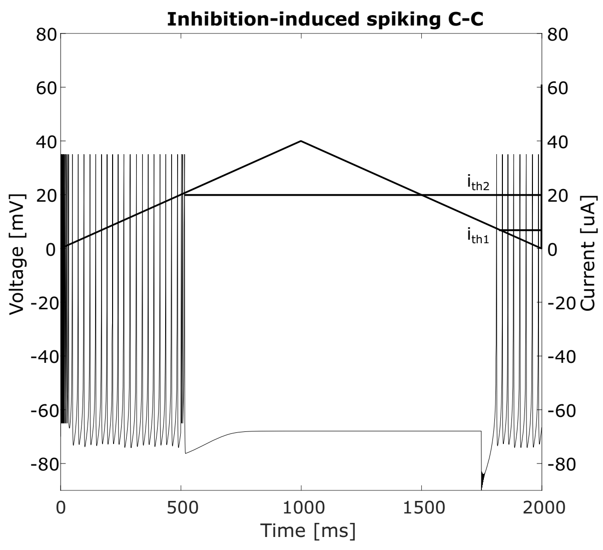

Fig. 6. Response of an inhibition-induced spiking neuron (IIS/C-C) based on a C-C model to a triangular excitation.

with it. The classifier either returns the correct answer (positive or negative) or no answer at all, suggesting the possible practical application of a 3-valued logic in the sense of Jan Łukasiewicz. Thanks to this, when using the SNN classifier as a pattern memory in the application of a self-learning agent, the hysteresis thresholds of synaptic current $i_{\text{th}1}, i_{\text{th}2}$ of the model defined with Eqn. (11) are activators for saving new patterns. The basis for the decision to save a new pattern in the classifier's memory is the detection of a unique feature, which activates the response of the C-C neuron. As mentioned, the complexity of the problem of evaluating the significance of features greatly exceeds the availability of computational resources of edge computing. However, the application of the three-state probability model leads to a situation when the condition for saving the new pattern in the classifier's memory is determining the uniqueness of features in relation to previously saved patterns. Thresholds $i_{\text{th}1}, i_{\text{th}2}$ are not variables in the learning process and are set to be identical for all features.

3. Thalamo-based architecture

The basis of the organization of neurons in individual layers of the SNN demonstrated in this paper is a phenomenon of lateral inhibition (Zhou *et al.*, 2022). It is one of the basic mechanisms of the functioning of nervous systems based on inhibiting the activity of adjacent neurons by stimulated neurons. This phenomenon is very desirable, because it leads to a selective response of the neural network, e.g., to selective, i.e., accurate mapping of stimulated nociceptors (pain receptors) (Quevedo *et al.*, 2017). Here, too, the cusp catastrophe model is used as a tool for modeling mental behavior, e.g., in the context of assessing suicidal behaviors induced by exposure to complex factors, including pain (de Beurs *et al.*, 2020). Both the functioning of nervous systems based on the lateral inhibition phenomenon, and the cusp catastrophe model assumes the production of a complex response in reply to a minor input function. In the discussed network both mechanisms are modeled using the aforementioned C-C neurons, the characteristics of which are shown in Figs. 5 and 6. Unlike the classical characteristics of IIS and TS neurons seen in Figs. 1 and 2, C-C models have a large margin of error. This is an important feature of the architecture, which makes it possible to apply learning methods without optimizing network parameters using iterative algorithms. The architecture of the SNN takes into account stimulating successive layers stimulated by signals generated for attributes (and not whole features) of the image. These signals act as stimuli to C-C neurons.

Regardless of the cusp catastrophe model, the tonic spiking and inhibition-induced spiking neurons are used in the network. As mentioned, this is inspired by the presence of both types of neurons in the thalamus, which acts as a preprocessor of stimuli before sending them to the cerebral cortex. The network has three layers according to the previously described network division: the *decomposition layer*, dividing the pattern into properties (attributes which do not directly define features), the *feature layer*, for cumulating properties into features and the *output layer*, cumulating features into groups of features, typical for each pattern.

In the process of network routing, dense connections were abandoned in favor of the feature aggregation

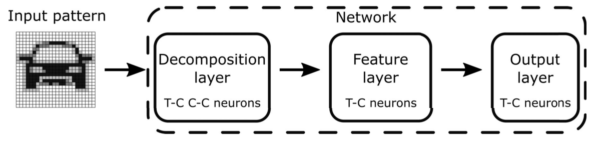

Fig. 7. Network architecture.

architecture in specific neurons. The details of this algorithm are described in Section 4. At this stage, however, it is worth mentioning that the average number of connections per neuron is more than 100 times smaller than for a typical dense connected network. Moreover, each first layer neuron is only connected to one second layer neuron. Such a strong reduction of connections can be treated as a pre-learning pruning and leads to homeostasis guaranteed by strong competition of neurons in approaches using lateral inhibition and the STDP learning method.

The basis for building a network responding to pattern attributes is to estimate the maximum number of synaptic connections for which the spiking neuron selectively classifies the origin of the stimulus. In other words, it is the maximum number of synapses to which a neuron responds when the signal changes at one of the synapses. It would be very difficult to achieve high selectivity based on C-C neurons alone. Limitations defined in Section 1, in particular the low dispersion of weights, additionally make it difficult to obtain high selectivity.

We solved this problem by using T-C neurons in the feature layer and in the output layer, i.e., neurons with spiking frequency dependent on the input current. This makes it possible to completely eliminate the dispersion of weights, i.e., to guarantee identical weights in all synapses in the entire network. This significantly reduces the cost of a hardware implementation of the network. Ultimately, it has been verified that the maximum number of neuron synapses in the decomposition layer for which the presented neuron models respond selectively when presenting an attribute on one of the synapses was determined experimentally and equals 70. The maximum number of connections per neuron is the basis for grouping network input signals. The grouping of information is the task of the second layer, i.e. the feature one. The task of the third layer, i.e., the output one, is to sum up information about the detected features. As in the case of the decomposition layer, T-C neurons are used in the other two layers due to the large amount of information coming from the previous layers.

A simplified diagram of the network is shown in Fig. 7. However, a detailed description of the network routing algorithm is provided in Section 4.

4. Learning by routing

Iterative SNN learning algorithms such as the STDP one are too expensive for applications in a self-learning agent. SNNs learned with the STDP algorithm are characterized by much larger sizes than, for example, networks learned with conversion algorithms, and the STDP algorithm itself is characterized by a much larger batch processing time (Pietrzak *et al.*, 2023). Iterative algorithms make it possible to obtain competitive parameters of classifiers, but in-situ training of the classifier cannot be done using a mobile GPU/FPGA-based system in real time. It is possible to implement STDP rules in hardware with the use of neuromorphic equipment (e.g., SpiNNaker (Diehl and Cook, 2014)); however, these solutions are very expensive and still not very popular. A limitation of classical algorithms is also the inability to train the existing network with new classes without a significant reconstruction of the network architecture.

A common drawback of learning multi-class problems using the STDP algorithm is the catastrophic forgetting phenomenon. This is a typical behavior of the classifier, which forgets previously learned patterns in the process of learning new patterns (Allred and Roy, 2020). For the above reasons, we proposed effective learning by a routing algorithm. The mechanism for creating a network described below is based on routing neurons based on image data, which leads to determining the network size and the architecture of connections between neurons in the network. We used an unsupervised method of self-learning, inspired by the methods of training associative memories such as Hebb's rule or analytical weight determination using the pseudo-inverse matrix.

Another difference in regard to STDP is using only strong connections. Iterative methods use mechanisms of strengthening or weakening synaptic connections, which leads to a class representation using weight values. However, it is also a source of dispersion of weight values. In our algorithm the perception is embedded in the connections themselves and not in the weights of these connections, which is the effect of the mechanism of efficient creation of strong binary connections with a low implementation cost. The network training scheme using M patterns represented by an image is shown in Fig. 8. Additionally, Algorithm 1 presents stages of the learning process in the form of a pseudocode. The required preprocessing of the image is its conversion to grayscale. That conversion does not affect quality as it is a popular approach in the widely described event-based vision processing (Gallego *et al.*, 2022). Additionally, a recommended but not obligatory process is histogram equalization. Each M pattern is composed of i pixels ($p_{1..i}$). In the first step of the routing algorithm, patterns are mapped to connections of a T-C+C-C neuron pair: one tonic spiking and one inhibition-induced spiking neuron with a cusp catastrophe mechanism for each pixel p in the pattern.

With these neurons, the spikes are sent to the feature layer, which maps the pattern's attributes on a catastrophic curve. Each attribute, depending on its position on the curve, is classified as active (the signal is transmitted by the tonic spiking C-C neuron) or inactive (the signal is transmitted by the inhibition-induced spiking C-C neuron). This classification is described in Algorithm 1 in Lines 4–10. Due to the dual characteristics of TS and IIS neurons and the hysteresis used in the C-C model, in each case only one neuron in a pair of neurons activates spiking. Whether or not the attribute will activate neurons of the next layer depends on exceeding the i-th hysteresis threshold. The next layer, the feature layer, aggregates attributes that make up features. Sample responses of a pair of C-C neurons for extreme values, i.e., with a maximum synaptic current (equivalent of the white pixel) and a minimum synaptic current (equivalent of the black pixel), are shown in Fig. 9.

In the subsequent stages of the learning algorithm, a subnetwork is created for each pattern $M_{1..k}$. For each pattern composed of i pixels, such a subnet consists of three layers. The first one is comprised of pairs of T-C+C-C neurons for each pixel of the pattern (in Fig. 8 these neurons are marked with symbol N). This part of the subnet is a component of the decomposition layer. For each M pattern, a matrix of subnet responses is generated based on neuron responses from each pair of N neurons. Based on this matrix, the structure of the second layer of the subnet consisting of T-C neurons is determined (in Fig. 8 these neurons are marked with the symbol F and are part of the feature layer). At this stage of the routing process, each F neuron has synapses connecting it with neurons of the decomposition layer which activated themselves for the given pattern. Thus, making connections is similar to learning rules by strengthening, but the network routing algorithm is not an iterative algorithm. According to the previous assumptions concerning selectivity, the maximum number of synaptic connections $F_{\max}$ in one neuron equals 70. The attribute aggregation algorithm selects successive neurons from the feature layer, provided that the maximum number of synaptic connections was reached in the neurons used so far. Creation of the second layer is described in Algorithm 1 in Lines 11–13.

In a situation where pattern classes have common features, their representation in the network is not duplicated (Algorithm 1, Lines 14–19). A given feature is represented by a constant number of neurons, even if the feature is common for different classes. This approach makes it possible to limit the network growth in the case of training subsequent classes, which increases the classifier's memory capacity.

The last stage of subnet routing for individual patterns is the selection of neurons from the output layer in which there is one neuron (O_k) collecting information from all neurons in the feature layer, which are active in the case of presentation of the training pattern M_i. Therefore, the last layer of the subnet for each pattern contains a single neuron from the output layer. The previously mentioned limitation of the maximum number of synaptic connections for this neuron does not exist anymore. The learning process ends with a clearing mechanism which clears the first layer of the network by removing unused neurons (Algorithm 1, Lines 30 to 34).

According to the previously adopted assumptions concerning low implementation costs, all synapses in the entire network, regardless of which layers they connect, have the same weight value equal to 0.0001.

The final stage of training is cleaning the network. This can be referred to as post-learning pruning. At this stage, unused neurons of the first layer are removed. Cleaning also checks that the created subnets represent common features. Such a situation occurs when patterns have the same attribute groups. The reduction in the network size is based on using one neuron in the feature layer for the same attribute groups. The cleaning process reduces the size of the previously trained network. The effectiveness of reduction depends on the number of classes (K). The cleaning process removes unconnected neurons from the first layer. The total number of neurons after the cleaning process is

$$n' = n \times (1 - 1.81^{-K}), \tag{12}$$

where n is the total number of neurons before the process.

Let us summarize the most important advantages of the presented algorithm. In contrast to iterative methods, our approach makes it possible to analytically route the network, similarly to methods used in classic associative memories. Thanks to this, it is possible to apply the algorithm in autonomous mobile devices with their strongly limited computing power, not enough to implement complex, biologically-inspired algorithms such as STDP. Our algorithm offers the possibility of training new patterns without having to retrain the entire network with previously remembered patterns. The algorithm is therefore resistant to the catastrophic forgetting problem. Similarly, new patterns representing already remembered classes can be used to train the network, i.e., to add connections to existing subnets. We offered a two-stage pruning of the network: before training at the connection creation stage and after training at the stage of eliminating redundant connections. Due to limited hardware resources of mobile systems, we propose a complete elimination of the weighting mechanism—all weights in the entire network have the same value. Despite the omission of the weight mechanism, the network has a large capacity, which is presented in detail in Section 5.

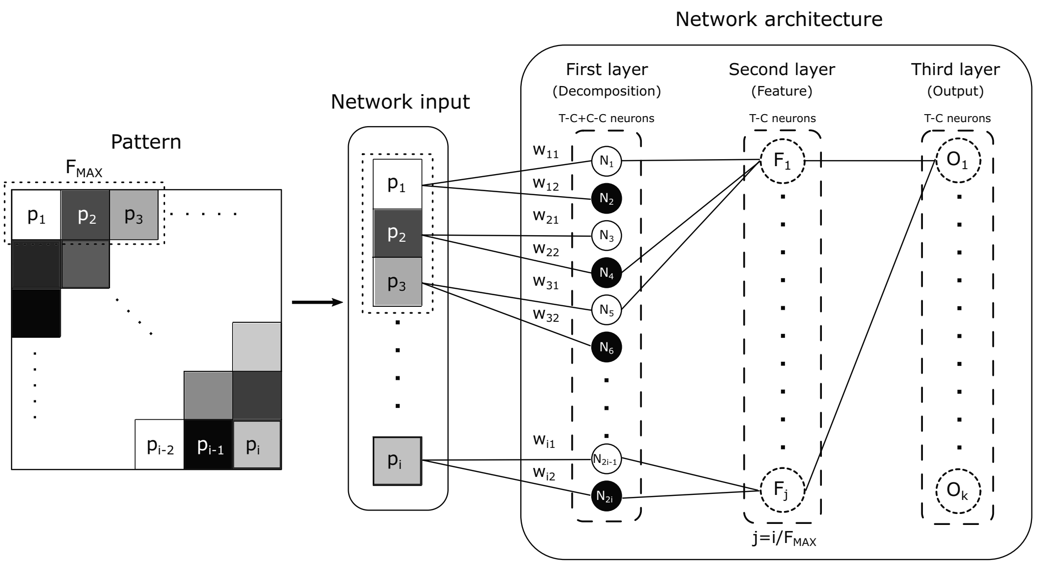

Fig. 8. Network routing diagram.

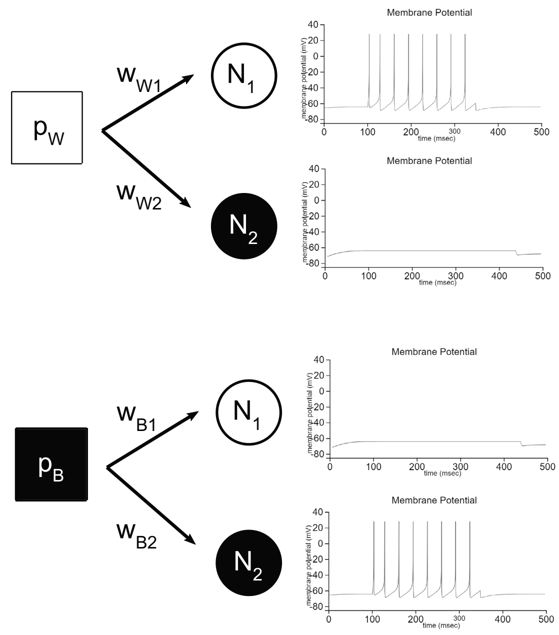

Fig. 9. Answer of a pair of neurons for each pixel of the image.

5. Experiments

To test how the network performs in a standalone agent application, we used the Caltech dataset 256 (Griffin *et al.*, 2007) containing examples of everyday objects. The photos were pre-processed: resized to the size of 70×70 pixels, the histogram equalized and converted to grayscale. The task of the network was to classify objects of the dataset. The network has an input size of 4900, and in the input layer at the beginning of the training algorithm there are 9800 neurons. Figure 10 presents example training patterns belonging to five different classes. Next to the patterns we show voltage signals generated at the network outputs. These signals are analyzed over a period of time T of 40 ms.

The traditional coding methods up to the first impulse turned out to be ineffective due to the network producing responses over a longer time horizon. Therefore, the coding method finally used is the average time of inactivity of the output t_{ina}, understood as the average distance $t_{i-1,i}$ between impulses $i-1$ and i in the analyzed time window according to

$$t_{\text{ina}} = \frac{1}{i}\sum_{i} t_{i-1,i}. \tag{13}$$

and Fig. 11.

The number of generated and analyzed impulses varies for different classes, but the neuron representing a given class is the most active, i.e., it has the shortest average inactivity time.

After training, without cleaning, the network has an architecture of $9800 - 350 - 5$ neurons in given layers, and after the cleaning it is $9615 - 350 - 5$. The architecture of the network changes but the number of connections remains the same. The average number of connections per neuron in the discussed network is 1.89. For a typical dense connections architecture, this value would be 253.97 with the same size of the input image. For the trained network, we used coding according to Eqn. (13). Figures 12–16 present example results of network operation for the test set created from the Caltech dataset.

The accuracy of the network at this stage of learning equals 87% and the precision for individual classes is for:

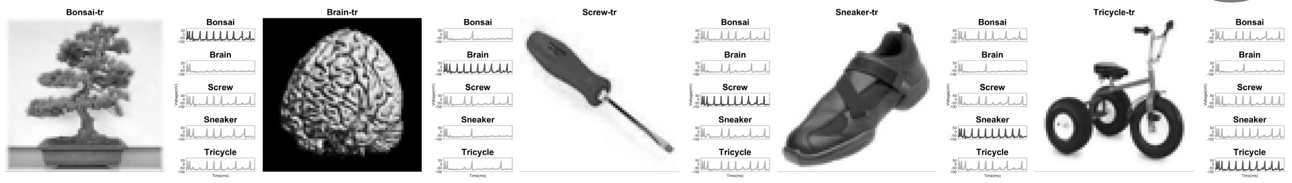

Fig. 10. Network training patterns.

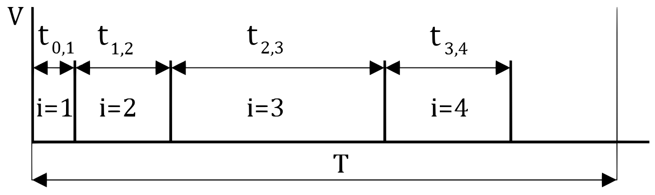

Fig. 11. Coding used in analyzing the activity of the classifier outputs.

Bonsai: 1, Brain: 0.9, Screw: 0.73, Sneaker: 1, Tricycle: 0.86.

Additionally, we conducted an analysis of network growth depending on the number of classes remembered by the network. The result of this analysis is shown in Fig. 17. The graph presents the size of the network determined in the simulation for real classification problems. The analysis presents the process of remembering subsequent classes, starting with a single one. The starting point on the graph (i.e., the number of neurons in the network for a single class) is the same for each set due to two features of the network creation algorithm: the cleaning of unused neurons in the first layer and aggregation of features common to different classes. Adding more classes to the network memory results in a slowing increase in the network complexity. This is caused by the operation of the mechanism of finding common features for classes and their non-duplication.

Due to the use of a training algorithm similar to associative memory learning methods, the network capacity can be determined depending on the number of neurons. The minimum number of neurons $S_{\min}$ needed to remember K classes of patterns of size x, with $F_{\max}$ connections per neuron in the hidden layer results from

$$S_{\min} = x \times \Big(1 + \frac{K}{2 \cdot F_{\max}}\Big) + K. \tag{14}$$

This is an impossible case, assuming that only x neurons were used in the hidden layer. This is a situation where each subsequent learning pattern differs by 50% from each previous grade. A difference below this threshold would mean presenting the next training pattern instead of the pattern representing the new class.

In turn, the maximum number of neurons results from

$$S_{\max} = x \times \Big(2 + \frac{K}{F_{\max}}\Big) + K. \tag{15}$$

This case assumes that twice as many neurons are used in the input layer, and the classes have no common features at all. For this situation, each successive class requires the use of $x \times K/F_{\max}$ neurons in the feature layer. This case is also impossible.

Both of the above functions are linear and divergent. This means that, as the number of classes increases, the width of the range between $S_{\max}$ and $S_{\min}$, which includes the actual dependence, increases linearly. The actual number of neurons needed to remember K classes depends on the type of classes. However, it can be assumed that, for complex problems, i.e., for a large number of classes, the number of classes that can be stored in memory is

$$K = \frac{1}{2}(S_{\max} + S_{\min}). \tag{16}$$

On the basis of tests of networks with sizes of 10k–500k neurons, we found that the network capacity can be estimated by the formula

$$K = 103 \times \frac{S}{x} - 138, \tag{17}$$

regardless of the training set. This dependency is valid for problems with the complexity of at least several dozen classes.

Finally, let us consider the complexity of the hardware implementation of the described algorithm. Table 1 summarizes the complexity of the network itself trained with various algorithms and using the Caltech dataset. The comparison is made for a five-class classifier. We focused on three main algorithms: STDP (supervised spike timing-dependent plasticity), event-based backpropagation SSTDP (supervised spike timing-dependent plasticity), which is characterized by a high efficiency of training classifiers (Pietrzak *et al.*, 2023), and ANN-SNN conversion, which is distinguished by a strong reduction in the complexity of the network architecture. The comparison shows the number of neurons, the number of synapses and the dispersion of synaptic connection weights.

As can be seen, our implementation has more neurons and fewer connections. To show how this

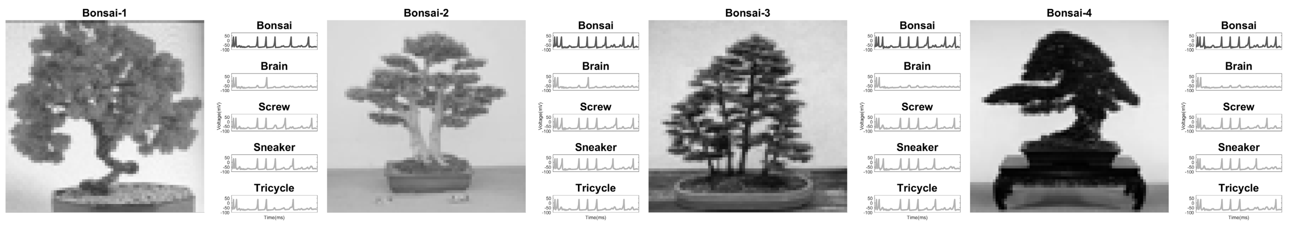

Fig. 12. Results of network classification for “Bonsai” patterns.

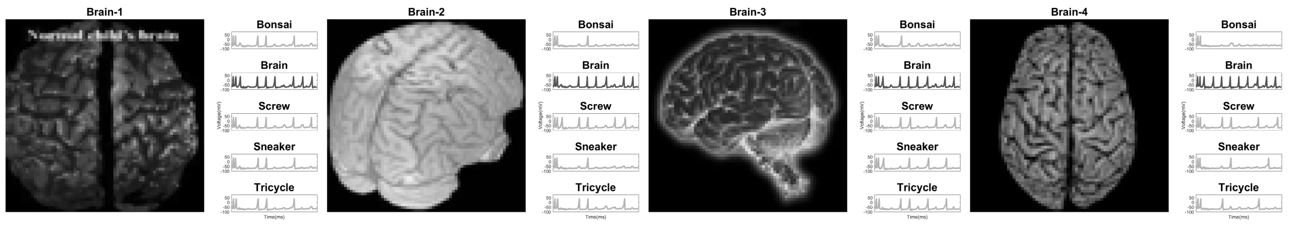

Fig. 13. Results of network classification for “Brain” patterns.

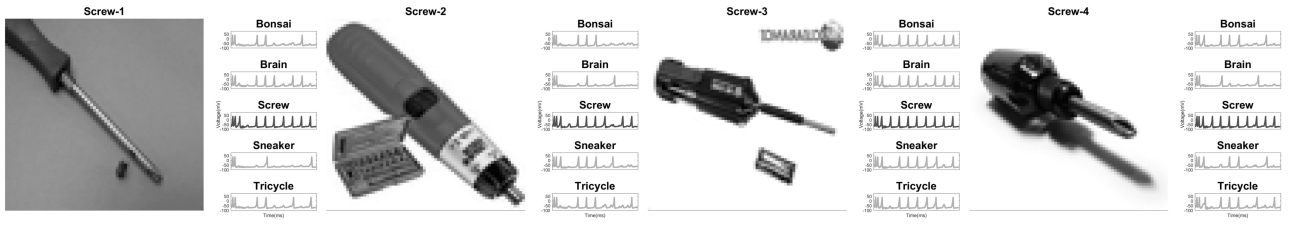

Fig. 14. Results of network classification for “Screw” patterns.

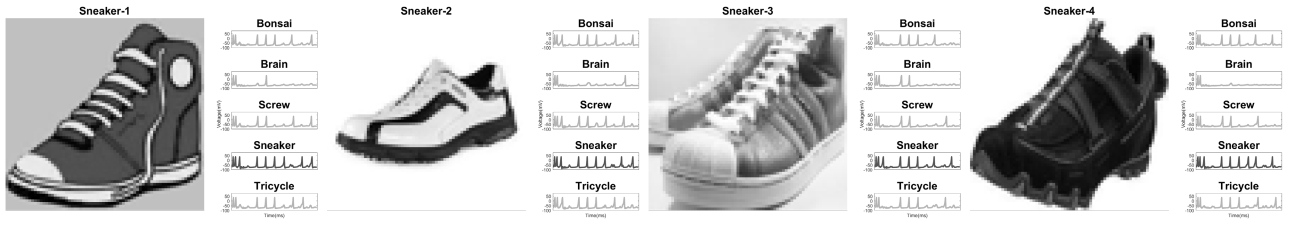

Fig. 15. Results of network classification for “Sneaker” patterns.

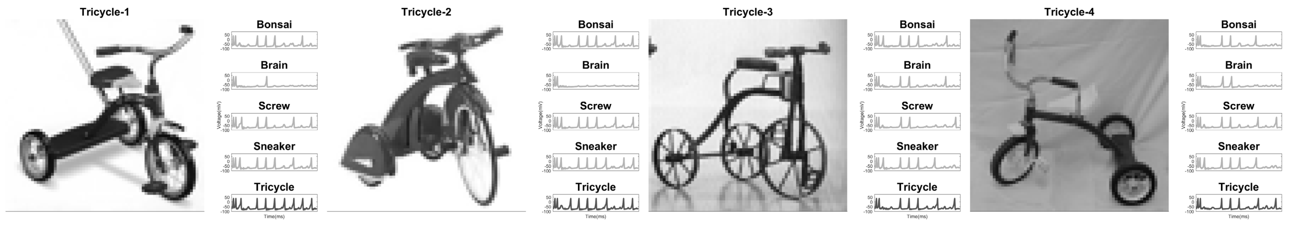

Fig. 16. Results of network classification for “Tricycle” patterns.

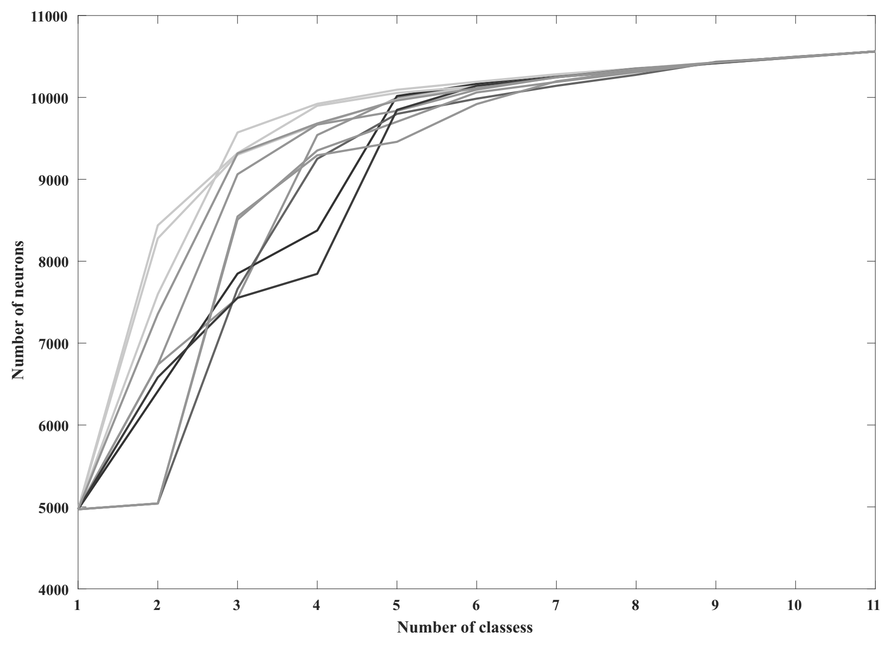

Fig. 17. Network growth analysis.

Table 1. Network architecture complexity for different learning algorithms.

	STDP	SSTDP	ANN-SNN conv	Our
Number of neurons	6 250 T-C	115 T-C	15 T-C	9 768 C-C, 353 T-C
Number of synapses	69 687 500	490 500	49 050	24 780
Weights dispersion	$[0, 741]$	$[-5.71, 10.488]$	$[-10, 10]$	0

affects the complexity of the hardware implementation, we made implementations using reconfigurable logic devices (FPGA). The analysis was made for the Xilinx 7 family equipped with 6-input LUTs. We considered two synthesis scenarios: using LUT blocks and dedicated DSP (digital signal processing) blocks to implement multiplication operations, and using only LUT blocks, i.e., implementing multiplications with their help. The results of the comparison are presented in Table 2. The individual rows of the table present the demand of the entire network for specific blocks. Note that our implementation in all cases is characterized by less use of hardware resources. Only in the case of using DSP blocks the LUT utilization was more than three times higher compared with the conversion algorithm. At the same time, the use of DSP blocks is over nine times smaller. The key parameter of FPGA accelerators, however, are DSP blocks, which affect the price of accelerators.

It should also be emphasized that the network implementation using the conversion algorithm is highly inefficient due to the naive way of generating the structure from a classical network and the use of frequency coding. This type of coding requires much more steps in the classifier simulation. The slow operation of the network after the conversion will be compounded by the slow hardware implementation. FPGA accelerators are usually equipped with several hundred DSP blocks. Only systems costing tens of thousands of dollars (e.g., Virtex 7) are equipped with about 5,000 DSP blocks. Higher use of DSP blocks requires employing memory to temporarily store the results. This further slows down the ANN-SNN conv implementation, where most DSP blocks are required.

6. Conclusion

In this work, we proposed an algorithm for training spiking neural networks dedicated to implementation in autonomous systems. The approach is strongly focused on reducing the complexity of both the learning algorithm and the final network. The presented solution, unlike other learning algorithms, is a noniterative approach, which enables in-situ learning, i.e., using an edge device. The advantages of the approach also include a strong reduction in the complexity of the SNN network itself through

Table 2. Hardware complexity of networks trained with different algorithms.

Training algorithm	STDP	SSTDP	ANN-SNN conv	Our
Using LUT and DSP blocks				
Number of LUT blocks	2 301 212.5k	16 206.77k	1 621.55k	5 177.32k
Number of DSP blocks	139 381.25k	981.11k	98.12k	10.12k
Using only LUT blocks				
Number of LUT blocks	21 885 043.75k	154 064.6k	15 408.5k	7 838.1k

the use of pre-learning and post-learning pruning and quantization of weights. Thanks to the approach modeled on associative memories, it is possible to estimate the required hardware resources for teaching subsequent classes. All these treatments translate into both lower energy costs of the final device and a larger capacity of the classifier itself. Additionally, the learned network is resistant to catastrophic forgetting.

References

Abusnaina, A. and Abdullah, R. (2014). Spiking neuron models: A review, *International Journal of Digital Content Technology and its Applications* **8**(3): 14–21.

Albert, Shalumov1, R., Halaly1, Elishai, E. and Tsur (2021). Lidar-driven spiking neural network for collision avoidance in autonomous driving, *Bioinspiration & Biomimetics* **16**(6): 066016.

Allred, J. and Roy, K. (2020). Controlled forgetting: Targeted stimulation and dopaminergic plasticity modulation for unsupervised lifelong learning in spiking neural networks, *Frontiers in Neuroscience* **14**: 1–16, Article no. 7.

Bartłomiejczyk, P., Trujillo, F.L. and Signerska-Rynkowska, J. (2023). Spike patterns and chaos in a map-based neuron model, *International Journal of Applied Mathematics and Computer Science* **33**(3): 395–408, DOI: 10.34768/amcs-2023-0028.

Cech, J., Hanis, T., Kononisky, A., Rurtle, T., Svancar, J. and Twardzik, T. (2021). Self-supervised learning of camera-based drivable surface roughness, *2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan*, pp. 1319–1325.

Chen, D.-G., Chen, X. and Zhang, K. (2016). An exploratory statistical cusp catastrophe model, *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, Canada*, pp. 100–109.

Chen, D.-G., Lin, F., Chen, X., Tang, W. and Kitzman, H. (2014). Cusp catastrophe model a nonlinear model for health outcomes in nursing research, *Nursing Research* **63**(3): 211–220.

Chen, X., Stanton, B., Chen, D.-G. and Li, X. (2013). Intention to use condom, cusp modeling, and evaluation of an HIV prevention intervention trial, *Nonlinear Dynamics, Psychology, and Life Sciences* **17**(3): 385–403.

Cheng, H.-P., Wen, W., Wu, C., Li, S., Li, H.H. and Chen, Y. (2017). Understanding the design of IBM neurosynaptic system and its tradeoffs: A user perspective, *Design, Automation & Test in Europe Conference & Exhibition (DATE 2017), Lausanne, Switzerland*, pp. 139–144.

Chu, L., Raghavendra, R., Srivatsa, M., Preece, A. and Harborne, D. (2019). Feature importance identification through bottleneck reconstruction, *2019 IEEE International Conference on Cognitive Computing (ICCC), Milan, Italy*, pp. 64–66.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.-K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y.-H., Wild, A., Yang, Y. and Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning, *IEEE Micro* **38**(1): 82–99.

Daw, R. and He, Z. (2020). Deep neural network in cusp catastrophe model, *arXiv:* 2004.02359, DOI: 10.48550/arXiv.2004.02359.

de Beurs, D., Bockting, C., Kerkhof, A., Scheepers, F., O'Connor, R. and van de Leemput, I. (2020). A network perspective on suicidal behavior: Understanding suicidality as a complex system, *Suicide and Life-Threatening Behavior* **51**(1): 115–126.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database, *2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA*, pp. 248–255.

Diehl, P. and Cook, M. (2014). Efficient implementation of STDP rules on spinnaker neuromorphic hardware, *Proceedings of the International Joint Conference on Neural Networks, Beijing, China,* pp. 4288–4295.

Encke, J. and Hemmert, W. (2018). Extraction of inter-aural time differences using a spiking neuron network model of the medial superior olive, *Frontiers in Neuroscience* **12**: 1–12, Article no. 140.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T. and Tian, Y. (2021). Incorporating learnable membrane time constant to enhance learning of spiking neural networks, *2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, Canada*, pp. 2641–2651.

Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K. and Scaramuzza, D. (2022). Event-based vision: A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**(1): 154–180.

Algorithm 1. Creating the network architecture.

```
Require: training patterns $tP$; maximal number of connections to feature $mCTF$; hysteresis level $hL$;
Ensure: network first layer (decomposition layer) $fL$; network second layer (features layer) $sL$; network third layer (output layer) $tL$;
 1: for $n = 1$ to count($tP$) do
 2:     $aTP = tP(n)$ /actual training pattern
 3:     $iV =$ toVector($aTP$) /input vector
        Decomposition layer output:
 4:     for $m = 1$ to $count(iV)$ do
 5:         if $iV(m) > hL$ then
 6:             $dLO = 1$
 7:         else
 8:             $dLO = 0$
 9:         end if
10:     end for
        Feature layer structure:
11:     for $j = 1$ to count($dLO == 1)/mCTF$ do
12:         $fLS(j) = dLO((j-1) \times mCTFj \times mCTF)$
13:     end for
        Check if feature already exist:
14:     for $k = 1$ to count($fLS$) do
15:         if $sL$.NotExist($fLS(k)$) then
16:             $sL$.Add($fLS(k)$)
17:         end if
18:     end for
19:     $tL(n) = fLS$
20: end for
    Assign features to correct patterns:
21: for $i = 1$ to count($tP$) do
22:     for $n = 1$ to count($sLN$) do
23:         if $tP(i)$.Exist($sLN(n)$) then
24:             $tP(i) += 1$
25:         else
26:             $tP(i) += 0$
27:         end if
28:     end for
29: end for
    Clear first layer:
30: for $i =$count($fL$) do
31:     if $sL$.NotExist($fL(i)$) then
32:         $fl$.Remove($fL(i)$)
33:     end if
34: end for
35: return $[fL, sL, tL]$
```

Griffin, G., Holub, A. and Perona, P. (2007). Caltech-256 object category dataset, *CalTech Report*, California Institute of Technology, Pasadena, DOI: 10.22002/D1.20087.

Guastello, S., Aruka, Y., Doyle, M. and Smerz, K. (2008). Cross-cultural generalizability of a cusp catastrophe model for binge drinking among college students, *Nonlinear Dynamics, Psychology, and Life Sciences* **12**(4): 397–407.

Halassa, M.M. and Acsády, L. (2016). Thalamic inhibition: Diverse sources, diverse scales, *Trends in Neurosciences* **39**(10): 680–693.

Hazan, H., Saunders, D., Sanghavi, D. T., Siegelmann, H. and Kozma, R. (2018). Unsupervised learning with self-organizing spiking neural networks, *2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil,* pp. 1–6.

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Deep residual learning for image recognition, *CoRR*: abs/1512.03385.

Hua, Y., Loomba, S., Pawlak, V., Voit, K.-M., Laserstein, P., Boergens, K.M., Wallace, D.J., Kerr, J.N. and Helmstaedter, M. (2022). Connectomic analysis of thalamus-driven disinhibition in cortical layer 4, *Cell Reports* **41**(2): 111476.

Huderek, D., Szczęsny, S. and Rato, R. (2019). Spiking neural network based on cusp catastrophe theory, *Foundations of Computing and Decision Sciences* **44**(3): 273–284.

Izhikevich, E. (2004). Which model to use for cortical spiking neurons?, *IEEE Transactions on Neural Networks* **15**(5): 1063–1070.

Kozdon, K. and Bentley, P. (2017). Wide learning: Using an ensemble of biologically-plausible spiking neural networks for unsupervised parallel classification of spatio-temporal patterns, *2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, USA*, pp. 1–8.

Leong, M., Prasad, D., Lee, Y. and Lin, F. (2020). Semi-CNN architecture for effective spatio-temporal learning in action recognition, *Applied Sciences* **10**(2): 557.

Li, G., Deng, L., Chua, Y., Li, P., Neftci, E.O. and Li, H. (2020). Editorial: Spiking neural network learning, benchmarking, programming and executing, *Frontiers in Neuroscience* **14**: 1–4, Article no. 276.

Li, H., Liu, H., Ji, X., Li, G. and Shi, L. (2017). CIFAR10-DVS: An event-stream dataset for object classification, *Frontiers in Neuroscience* **11**: 1–10, Article no. 309.

Liang, Q., Shenoy, P. and Irwin, D. (2020). AI on the edge: Characterizing AI-based IoT applications using specialized edge architectures, *2020 IEEE International Symposium on Workload Characterization (IISWC), Beijing, China*, pp. 145–156.

Lim, Y. and Golden, J.A. (2007). Patterning the developing diencephalon, *Brain Research Reviews* **53**(1): 17–26.

Markram, H., Gerstner, W. and Sjöström, P.J. (2012). Spike-timing-dependent plasticity: A comprehensive overview, *Frontiers in Synaptic Neuroscience* **4**: 1–3, Article no. 2.

Mayr, C., Hoeppner, S. and Furber, S. (2019). Spinnaker 2: A 10 million core processor system for brain simulation and machine learning, *arXiv:* 1911.02385, DOI: 10.48550/arXiv.1911.02385.

Meftah, B., Lezoray, O., Lecluse, M. and Benyettou, A. (2010). Cell microscopic segmentation with spiking neuron networks, *in* K. Diamantaras *et al.* (Eds), *Artificial Neural Networks—ICANN 2010*, Springer, Berlin/Heidelberg, pp. 117–126.

Na, B., Mok, J., Park, S., Lee, D., Choe, H. and Yoon, S. (2022). AutoSNN: Towards energy-efficient spiking neural networks, *39th International Conference on Machine Learning, Baltimore, USA*.

Nazari, S., Amiri, M., Faez, K. and Van Hulle, M.M. (2020). Information transmitted from bioinspired neuron–astrocyte network improves cortical spiking network's pattern recognition performance, *IEEE Transactions on Neural Networks and Learning Systems* **31**(2): 464–474.

Neculae, G. (2020). *Ensemble Learning for Spiking Neural Networks*, PhD thesis, University of Manchester, Manchester.

Neftci, E.O., Mostafa, H. and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks, *IEEE Signal Processing Magazine* **36**(6): 51–63.

Oster, S., Deiner, M., Birgbauer, E. and Sretavan, D. (2004). Ganglion cell axon pathfinding in the retina and optic nerve, *Seminars in Cell & Developmental Biology* **15**(1): 125–136.

Pereira-Pires, J.E., Ferreira, J. and Rato, R. (2019). Spike based computing: A novel hardware to compute and control with spikes in space, ESA contract nº 4000117067/16/NL/MH/gm, *Final Report*, European Space Agency, Paris, DOI: 10.13140/RG.2.2.22848.97286.

Pietrzak, P., Szczęsny, S., Huderek, D. and Przyborowski, L. (2023). Overview of spiking neural network learning approaches and their computational complexities, *Sensors* **23**(6): 3037.

Pisarev, A., Busygin, A., Udovichenko, S.Y. and Maevsky, O. (2020). A biomorphic neuroprocessor based on a composite memristor-diode crossbar, *Microelectronics Journal* **102**: 104827, DOI: 10.1016/j.mejo.2020.104827.

Ponghiran, W., Srinivasan, G. and Roy, K. (2019). Reinforcement learning with low-complexity liquid state machines, *Frontiers in Neuroscience* **13**: 1–14, Article no. 883.

Ponulak, F. (2008). Analysis of the ReSuMe learning process for spiking neural networks, *International Journal of Applied Mathematics and Computer Science* **18**(2): 117–127, DOI: 10.2478/v10006-008-0011-1.

Quevedo, A., Mørch, C., Andersen, O. and Coghill, R. (2017). Lateral inhibition during nociceptive processing, *PAIN* **158**(6): 1046–1052.

Raha, A., Kim, S.K., Mathaikutty, D.A., Venkataramanan, G., Mohapatra, D., Sung, R., Brick, C. and Chinya, G.N. (2021). Design considerations for edge neural network accelerators: An industry perspective, *2021 34th International Conference on VLSI Design/2021 20th International Conference on Embedded Systems (VLSID), Guwahati, India*, pp. 328–333.

Rajbahadur, G.K., Wang, S., Oliva, G.A., Kamei, Y. and Hassan, A.E. (2022). The impact of feature importance methods on the interpretation of defect classifiers, *IEEE Transactions on Software Engineering* **48**(7): 2245–2261.

Raz, Halaly, Elishai, E. and Tsur (2023). Autonomous driving controllers with neuromorphic spiking neural networks, *Front Neurorobot* **17**: 1234962, DOI: 10.3389/fnbot.2023.1234962.

Rowcliffe, P., Feng, J. and Buxton, H. (2006). Spiking perceptrons, *IEEE Transactions on Neural Networks* **17**(3): 803–807.

Salt, L., Howard, D., Indiveri, G. and Sandamirskaya, Y. (2020). Parameter optimization and learning in a spiking neural network for UAV obstacle avoidance targeting neuromorphic processors, *IEEE Transactions on Neural Networks and Learning Systems* **31**(9): 3305–3318.

Shrestha, Sumit, B. and Orchard, G. (2018). Slayer: Spike layer error reassignment in time, *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canada*.

Stagsted, R., Vitale, A., Binz, J., Renner, A., Larsen, L.B. and Sandamirskaya, Y. (2020). Towards neuromorphic control: A spiking neural network based PID controller for UAV, *Robotics: Science and Systems 2020, Corvalis, USA*, pp. 1–8, DOI: 10.5167/uzh-200415.

Szczęsny, S. (2017). 0.3 V 2.5 nW per channel current-mode CMOS perceptron for biomedical signal processing in amperometry, *IEEE Sensors Journal* **17**(17): 5399–5409.

Szczęsny, S., Huderek, D. and Przyborowski, L. (2021). Spiking neural network with linear computational complexity for waveform analysis in amperometry, *Sensors* **21**(9): 3276.

Tazerart, S., Mitchell, D.E., Miranda-Rottmann, S. and Araya, R. (2019). A spike-timing-dependent plasticity rule for single, clustered and distributed dendritic spines, *bioRxiv*: 397323, `https://www.biorxiv.org/content/early/2019/01/27/397323`.

Torrico, T. and Munakomi, S. (2020). *Neuroanatomy, Thalamus,* National Library of Medicine, Bethesda, `https://www.ncbi.nlm.nih.gov/books/NBK542184/`.

Viale, A., Marchisio, A., Martina, M., Masera, G. and Shafique, M. (2021). CARSNN: An efficient spiking neural network for event-based autonomous cars on the Loihi neuromorphic research processor, *2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China,* pp. 1–10.

Vosahlik, D., Cech, J., Hanis, T., Konopisky, A., Rurtle, T., Svancar, J. and Twardzik, T. (2021). Self-supervised learning of camera-based drivable surface friction, *2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, USA*, pp. 2773–2780.

Wu, D., Yi, X. and Huang, X. (2022). A little energy goes a long way: Build an energy-efficient, accurate spiking neural network from convolutional neural network, *Frontiers in Neuroscience* **16**: 1–11, Article no. 759900.

Wu, Y., Deng, L., Li, G., Zhu, J. and Shi, L. (2018a). Direct training for spiking neural networks: Faster, larger, better, *3rd AAAI Conference on Artificial Intelligence (AAAI-19), Honolulu, USA,* pp. 1311–1318.

Wu, Y., Deng, L., Li, G., Zhu, J. and Shi, L. (2018b). Spatio-temporal backpropagation for training high-performance spiking neural networks, *Frontiers in Neuroscience* **12**: 1–12, Article no. 331.

Xuelei, C. (2023). Autonomous driving using spiking neural networks on dynamic vision sensor data: A case study of traffic light change detection, *arXiv*: 2311.09225, DOI: 10.48550/arXiv.2311.09225.

Zhao, J., Fang, J., Ye, Z. and Zhang, L. (2021). Large scale autonomous driving scenarios clustering with self-supervised feature extraction, *2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan,* pp. 473–480.

Zhou, J., Dai, J. and Weng, S. (2022). Effect of adjacent lateral inhibition on light and electric-stimulated synaptic transistors, *IEEE Electron Device Letters* **43**(4): 573–575.

Damian Huderek graduated in automatic control and robotics from the Faculty of Computer Science of the Poznań University of Technology with an MS degree in 2017. Then, he began doctoral studies in the field of computer science there. His main research interests include bio-inspired neural networks and their application, programming and PCB design.

Szymon Szczęsny graduated in 2008 in automation and management from the Faculty of Computing and Management of the Poznan University of Technology. In 2013 he received his PhD in computer science there. He is mainly interested in bio-inspired electronics, computational neuroscience and processing bio-medical signals. He also works on tasks of automating processes of designing layouts of current-mode circuits, prediction of defects in processes of topography fabrication and algorithms for synthesizing analog circuits by using hardware description languages.

Paweł Pietrzak received his MS degree in computer science from the Faculty of Computing and Telecommunications of the Poznań University of Technology in 2020. In the same year he became a PhD student there and started working full time as a machine learning engineer in the field of medical imaging software. His main interests include artificial intelligence, computer vision and embedded systems.

Raul Rato is a professor responsible for signal analysis at the NOVA School of Science and Technology. With a career spanning over three decades, he has contributed to the field with numerous papers. Doctor Rato's research centers on harnessing signals as sub-symbolic vehicles for computational tasks. Presently, his pioneering work explores the direct utilization of signals as carriers of information in computing, bypassing the need for analog-to-digital conversion. This innovative approach holds the promise of revolutionizing physical-level communication security and enabling wave-based computing at hyper frequencies.

Łukasz Przyborowski graduated in mechatronics from the Faculty of Mechanical Engineering and Management of the Poznań University of Technology in 2014. In 2017 he received his MS degree in computer science from the Faculty of Computer Science of that same university, where he started PhD studies in 2019 in the field of artificial neural networks based on pulsed neurons. He is interested in programming and now professionally programs web applications.

Received: 6 November 2023
Revised: 29 April 2024
Accepted: 9 June 2024