WiN-GUI Version 2: A graphical tool for neuron-based encoding

SoftwareX 29 (2025) 102031

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Software update

WiN-GUI Version 2: A graphical tool for neuron-based encoding

Vittorio Fra [a,*], Simon F. Müller-Cleve [b], Gianvito Urgese [a], Chiara Bartolozzi [b]

[a] Politecnico di Torino, Turin, Italy
[b] EDPR, Istituto Italiano di Tecnologia, Genoa, Italy

ARTICLE INFO

Keywords:
Neuromorphic
Spike encoding
Development tool
Spike-pattern classification
Robotics
Graphical user interface

ABSTRACT

The WiN-GUI enables real-time exploration of neuron model behaviors by adjusting internal parameters and accounting for input properties such as scale and temporal resolution. The spiking responses are however often difficult to interpret and categorize. The update to version 2 introduces a systematic labeling of spike-patterns to facilitate clearer communication across researchers and disciplines, enabling a common framework to describe neuron responses without sharing data. Labels also allow researchers to intentionally target specific neuronal behaviors, fostering biologically plausible simulations or specific tuning goals. To this end, our WiN-GUI incorporates a spike-pattern classifier for automated identification and analysis of neuron activity, streamlining research and collaboration.

Code metadata

Current code version	v2.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00602
Code Ocean compute capsule	https://github.com/event-driven-robotics/win-GUI/releases/tag/v2.0
Legal Code License	GPL-3.0
Code versioning system used	git
Software code languages, tools, and services used	Matplotlib, Pandas, Numpy, PyDub, PyQt6, Python, PyTorch, scikit-learn, SciPy
Compilation requirements, operating environments & dependencies	Requires PyQT6 and PyTorch with Python. Tested on Linux and Windows
If available Link to developer documentation/manual	
Support email for questions	vittorio.fra@polito.it

Description of the software-update

The WiN-GUI [1] is designed to develop spike encoding schemes for sample-based data. It supports multiple neuron models and allows dynamic adjustments to input dynamics and neuron parameters, providing immediate observation of their effects on neuronal responses. Multi-channel input and internal state variables are visualized using a unified color-coding scheme for clarity.

Biological neurons display diverse spiking behaviors, extensively documented in the literature, e.g. in [2] which categorizes 20 distinct patterns. These patterns, expanded and illustrated in Fig. 1, serve as a foundation for classification. To improve and standardize communication among researchers regarding spiking activity influenced by input or parameter variations, we enhanced WiN-GUI to classify encoding strategies based on these patterns.

Such classification employs the Recurrent Spiking Neural Network (RSNN) model described in [3], trained on the 20 above mentioned spiking behaviors. The tool can also simplify results by displaying only five principal categories: *Regular* (periodic single spikes), *Single burst* (a single spike burst), *Multi-burst* (multiple spike bursts), *Mixed* (variable Interspike Interval (ISI) patterns), and *Unstructured* (indistinct features lacking classification).

1. Spike-pattern classifier integration

The original content of the WiN-GUI is now located in the "Data Visualization" tab and remains unchanged. The new "Spike-Pattern Visualization" tab presents the results produced by means of a spike-pattern classifier.

DOI of original article: https://doi.org/10.1016/j.softx.2024.101759.
* Corresponding author.
E-mail address: vittorio.fra@polito.it (Vittorio Fra).

https://doi.org/10.1016/j.softx.2024.102031
Received 11 November 2024; Received in revised form 20 December 2024; Accepted 22 December 2024
Available online 13 January 2025

Within this tab, the first column lists input sensor IDs. The second column provides the predicted spike pattern, while subsequent columns present class probabilities. The latter indicate the distribution and classification confidence, and they are displayed as percentages in color-coded cells. If no spikes are detected, the row remains empty with white cells. Classification can be performed and visualized with different levels of detail, whose selection is possible through checkboxes.

The batch size for classification defaults to 128 but it can be modified in the code. For channels with multiple trials from windowing, the mean likelihood determines the winning class. Adjusting neuron parameters with the sliders or modifying input properties with checkboxes initiates the classification process, which is reported and updated upon selection of the "Pattern classification" checkbox. A terminal progress bar displays its status.

Section 2 details the dataset preparation for the spike-pattern classifier training, including the definition of behavioral super-classes. Section 3 covers the classifier's model definition, training, and optimization. Finally, Section 4 provides an example of spike-pattern classification using the "Spike-Pattern Visualization" tab, and Section 5 discusses current limitations and potential improvements.

2. Spike-pattern dataset

In the work by Mihalaş and Niebur (2009) [2], 20 distinct neuronal behaviors are documented. We replicated these behaviors using a PyTorch model as described in [1], adhering to the equations and parameters specified in [2]. By introducing noise and jitter, we generated a dataset with high variance yet retained separability.

Unlike Mihalaş and Niebur [2], we explicitly consider C as an internal neuron parameter rather than a scaling factor, thus the neuronal response is driven by the excitatory input I_e instead of the original I_e/C. The resulting behavior is then influenced by the neuron parameters, the input strength and dynamics, and the initial conditions. The table presented in [2] distinguishes spiking behaviors with identical neuron parameters by grouping them with horizontal lines. Input characteristics I_e predominantly dictate the neuronal activity, with two exceptions requiring specific initial conditions.

The most distinct spiking behaviors occur with $a = 5\ \mathrm{s}^{-1}$, $A_1 = 0$ A, and $A_2 = 0$ A under various inputs (Fig. 1, C–J). Bursting responses are elicited with $A_1 = 10$ A and $A_2 = -0.6$ A (Fig. 1, M–O). For constant input $I_e = 1.5$ A, we observe *Tonic spiking*, *Phasic spiking*, and *Phasic bursting* (Fig. 1, A, D, N). With temporal dynamics (step input), *Integrator* emerges (Fig. 1, I). Similarly, with $I_e = 2$ A, we observe *Spike frequency adaptation*, *Tonic bursting*, and *Mixed mode*; adding a step results in *Afterpotential* (Fig. 1, C, M, P, Q). For *Class 1*, the potential E_L must be set to the threshold Θ, with Θ_∞ just below -0.0500002 V, as in *Mihalaş and Niebur*. To elicit *Class 2*, E_L and Θ must be initialized at -0.03 V (Fig. 1, B, H).

The dataset is produced with trials of 1 s and a simulation time step of 1 ms. Compared to [2], I_e is perturbed via: (I) addition of white noise η_{noise}, (II) a random constant offset μ_{offset}, and (III) random temporal jitter for step inputs (τ_{jitter}). We used the Silhouette score of mean ISIs to assess noise levels. The score starts at 0.72 with good separability and decays as noise increases. Noise (η_{noise}) impacted decay more than offset (μ_{offset}). Noise and offset levels beyond 0.2 caused high-frequency spiking, reducing class distinction (Silhouette score 0.1). To account for variability, we generated 100 trials per class and noise combination using:

(I) $\eta_{noise} = 0.1$ V/s and $\eta_{noise} = 0.2$ V/s,
(I) $\mu_{offset} \sim U(-0.1, 0.1)$ V/s and $\mu_{offset} \sim U(-0.2, 0.2)$ V/s,
(I) $\tau_{jitter} \sim U(-10, 10)$ ms.

With respect to the original data [2], most features remain consistent, though perturbations primarily affect features originally one-second long. Some behaviors repeat characteristic patterns, depending on trial length. In *Class 1*, noise relaxes the threshold condition, increasing firing rate by frequently pushing the membrane potential above the threshold. In *Afterpotentials*, noise sometimes prevents the membrane from reaching threshold. The most noticeable change occurs in *Basal bistability* (Fig. 1, R), where bursting is influenced by noise, offset, and jitter.

Behaviors super-class definition

For the purposes of WiN-GUI, and specifically in this update, we introduce an additional, higher level of analysis for neuronal behaviors. Expanding on the spiking activity set presented in [2], we define five super-classes to group behaviors with similar features. To establish a systematic labeling that simplifies the design of spike-encoding modules, a coarse-grained classification is indeed advantageous. This captures key characteristics common to different behaviors, enabling further refinement through a two-step approach able to target the most appropriate class for specific goals. Conversely, super-classes alone can be useful in exploratory phases where fine discrimination is unnecessary and a macroscopic view of behaviors suffices.

The super-classes are as follows: *Regular*, for patterns with periodic single spikes; *Single burst*, for responses that generate a single spike burst; *Multi-burst*, for behaviors with multiple bursts; *Mixed*, for activities with variable ISI; and *Unstructured*, for cases without clear defining features from the categories above.

The neuronal behaviors illustrated in Fig. 1 are grouped by super-class as follows: A, B, K, and Q in *Regular*; *N* and O in *Single burst*; L, M, R, and S in *Multi-burst*; C, D, E, H, J, and P in *Mixed*; F, G, I, and *T* in *Unstructured*.

3. Spike-pattern classifier

Model definition

A two-layer fully connected RSNN, inspired by *Müller-Cleve et al.* (2022) [3], was utilized, featuring a virtual input layer and Current-Based LIF (CuBa-LIF) neurons in both the hidden recurrent and the output layer. The discretized neuron model is defined by the following equations for the ith neuron:

$$I_i[\mathrm{t}] = \alpha I_i[\mathrm{t}-1] + I_{in,i}[\mathrm{t}] + \sum_j V_{ij} \cdot S_j[\mathrm{t}-1]$$

$$U_i[\mathrm{t}] = (\beta U_i[\mathrm{t}-1] + I_i[\mathrm{t}]) \cdot (1 - S_i[\mathrm{t}-1])$$

where I_i and U_i represent the synaptic current and the membrane potential respectively; $I_{in,i}[\mathrm{t}] = \sum_j W_{ij} \cdot S_j[\mathrm{t}-1]$ represents the weighted spiking activity from the previous layer; $\alpha = \exp\left(\frac{-1}{\tau_{syn}}\right)$ and $\beta = \exp\left(\frac{-1}{\tau_{mem}}\right)$ are decay constants for the synaptic current and membrane potential, respectively; $\sum_j V_{ij} \cdot S_j[\mathrm{t}]$ is the recurrent term, $S_j[\mathrm{t}] = \{0, 1\}$ indicates neuron firing; and $(1 - S_i[\mathrm{t}-1])$ models the hard reset mechanism for neuron i upon firing.

The network was trained using Backpropagation Through Time (BPTT) with supervised learning through cross-entropy on the output layer activity, and a surrogate gradient was applied, modeled by a fast sigmoid as follows:

$$\sigma(U_i) = \frac{U_i}{1 + \lambda |U_i|}$$

with λ being a hyperparameter set to 10 for the present model.

To address potential pathological firing behaviors within the network [4], two regularization terms[1] were added to the hidden recurrent layer, resulting in the overall loss function defined as:

$$\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_1 + \mathcal{L}_2$$

[1] https://github.com/fzenke/spytorch

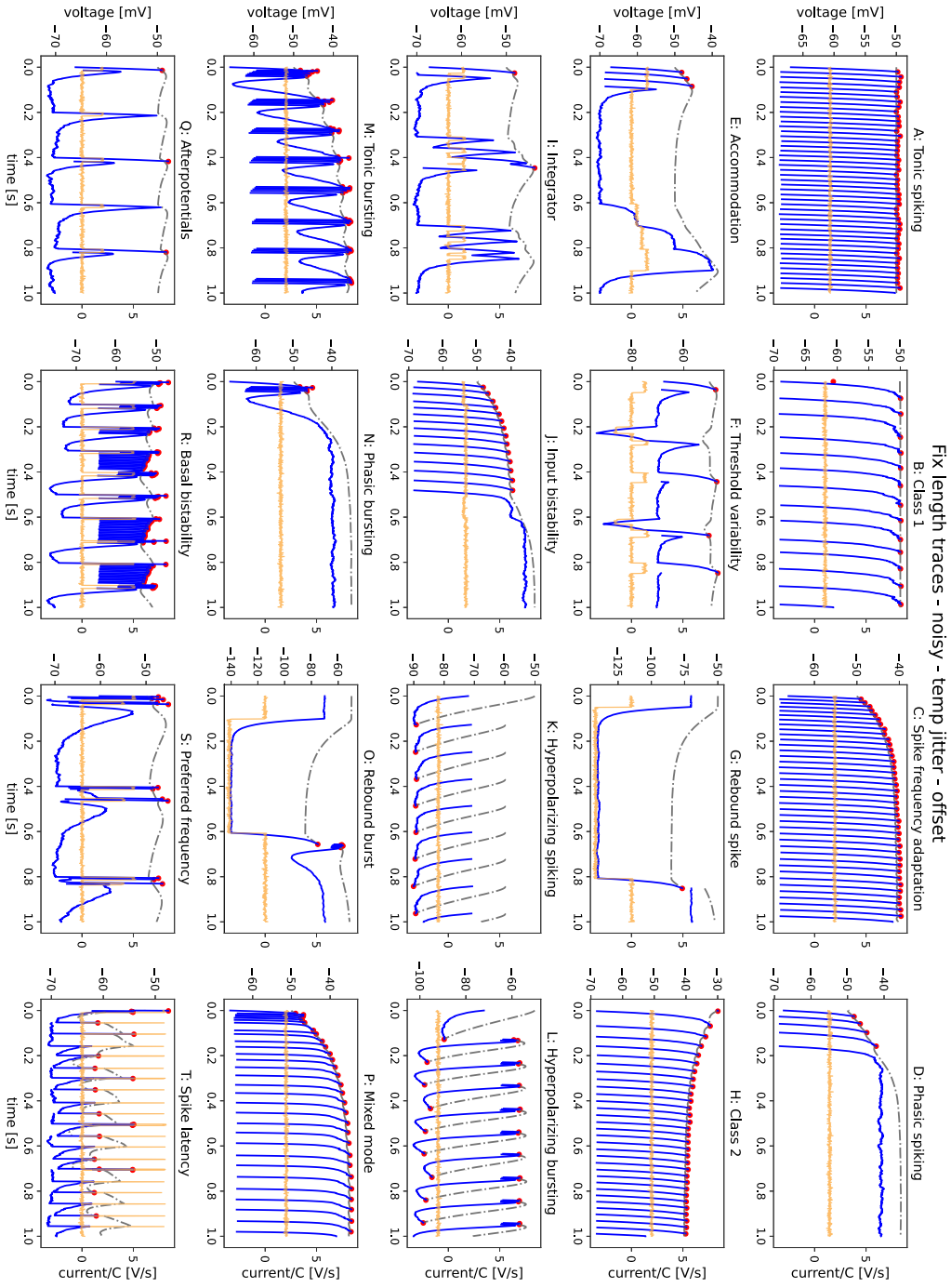


Fig. 1. Representative samples from the spike-pattern dataset were generated by introducing white noise and jitter into the neural responses from [2]. The orange line denotes the input current, the blue line represents the neuron's membrane potential, and the red dots indicate spike events. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\mathcal{L}_{ce} = -\frac{1}{B}\sum_{s=1}^{B} \mathbb{1}(i = y_s)\cdot \cdot \log\left\{\frac{\exp\left(\sum_{n=1}^{T} S_i^{(l)}[n]\right)}{\sum_{i=1}^{N_{class}} \exp\left(\sum_{n=1}^{T} S_i^{(l)}[n]\right)}\right\}$$

$$\mathcal{L}_1 = \frac{\mu_1}{B \cdot N}\sum_{B}\sum_{N}\left(\sum_{n=1}^{T} S_i[n]\right)$$

$$\mathcal{L}_2 = \frac{\mu_2}{B \cdot N}\sum_{N}\left(\sum_{B}\sum_{n=1}^{T} S_i[n]\right)^2$$

where B is the batch size, N is the number of neurons in the hidden recurrent layer, S_i is the spiking activity of the i-the neuron in this layer, and the strength coefficients 1×10^{-4} and 1×10^{-8}, respectively.

Hyperparameter optimization

We performed HPO for the RSNN-based spike-pattern classifier using the Neural Network Intelligence (NNI) toolkit, conducting one thousand training trials with validation at each epoch, following a

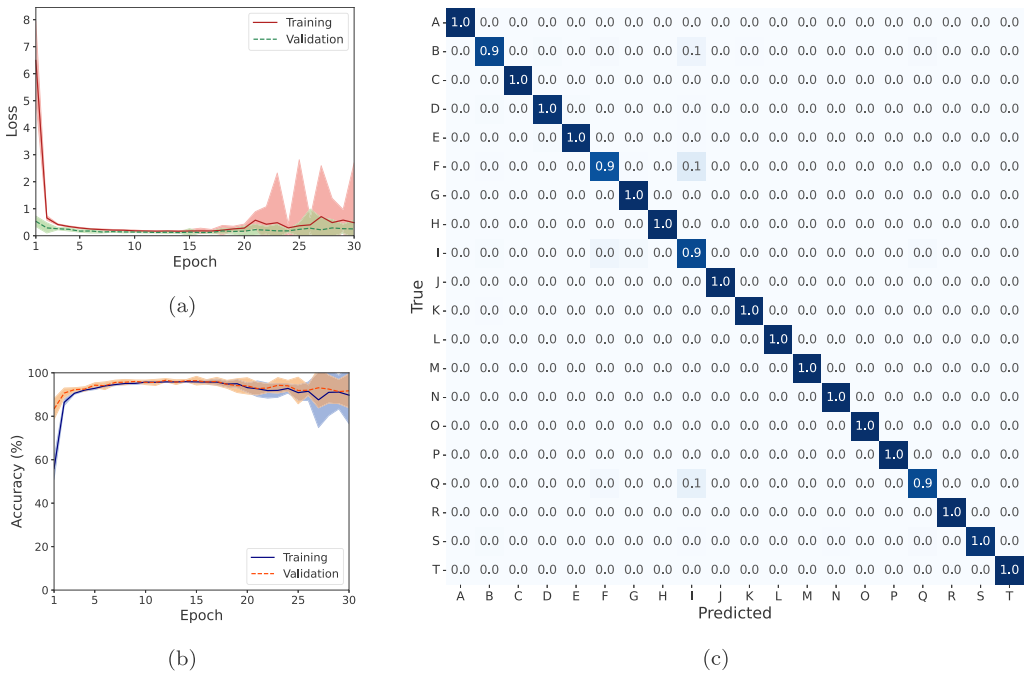


Fig. 2. The learning curves from retraining the best model after the HPO experiment are presented in (a) and (b), showing the statistical analysis of loss and accuracy respectively. Solid lines represent median values, while shaded areas indicate standard deviations. In (c), the confusion matrix for the test set is displayed.

procedure adapted from *Fra et al.* (2022) [5]. The built-in *Anneal* tuner was used, with random reinitialization every 200 trials to mitigate local minima effects [6]; and the highest validation accuracy during training served as the objective metrics, with the optimal model selected based on test accuracy.

The spike-pattern dataset described above was divided into training, validation, and test set with a 70:20:10 ratio.

Optimal classifier

Once the optimal hyperparameters were identified, we conducted ten additional training repetitions with random training-validation subsets and a fixed test set. Training for less than 30 epochs proved sufficient for achieving peak performance, with early forgetting observed after approximately 20 epochs as reported in Fig. 2(a) and Fig. 2(b).

The final model, selected for the highest test accuracy among those with the best validation accuracy in each repetition, provided classification on the test set with accuracy equal to 97.58% and the confusion matrix reported in Fig. 2(c). Such optimal model was then evaluated on the super-classes as well, by performing classification of the test set with super-class labels. The resulting confusion matrix is shown in Fig. 3, corresponding to an overall accuracy of 98.78%.

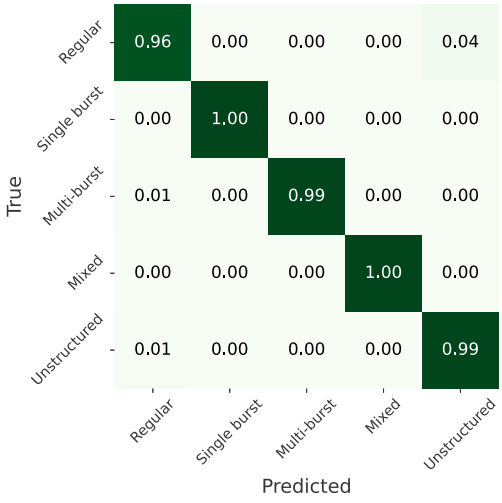


Fig. 3. Confusion matrix generated by the optimal model on the test set, focusing on super-classes, with an overall accuracy of 98.78%.

4. Illustrative examples

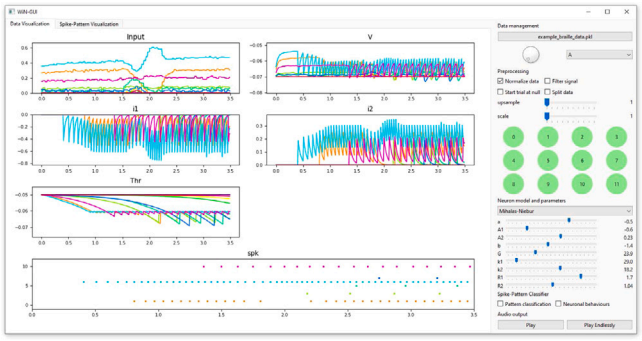
The visualizations provided by WiN-GUI are shown in Fig. 4, using input data from the tactile Braille reading dataset.[2]

The "Data Visualization" tab, illustrated in Fig. 4(a), presents primary features. Graphic panels are on the left-hand side, while the right-hand side contains a dashboard for data import and preprocessing, neuron selection, and configuration. Displayed quantities include input signals, internal neuronal states and spiking activity. Compared to the previous WiN-GUI version, the "Recurrent integrate-and-fire" neuron model has been replaced with the "Current-based integrate-and-fire" neuron, and checkboxes for the classification of spiking activity can now be found under the "Spike-Pattern Classifier" section, directly above the audio controls.

In Fig. 4(b), the "Spike-Pattern Visualization" tab displays classification results when only the "Pattern classification" checkbox is selected. With this option enabled, spike-pattern classification is conducted at the super-class level. When the "Neuronal behaviors" checkbox is also selected, an extended classification is displayed, as shown in Fig. 4(c).

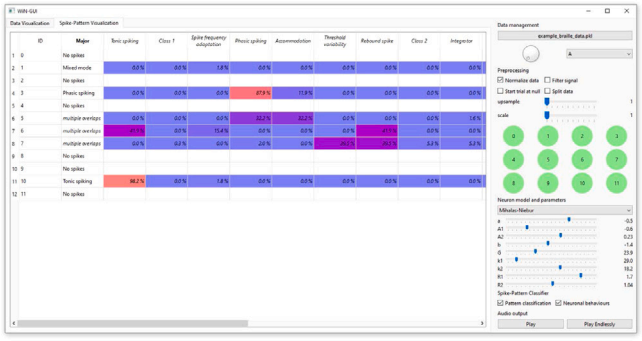
In both Fig. 4(b) and Fig. 4(c), WiN-GUI provides classification results as probabilities for each input channel. The "Major" column summarizes the predominant behavior identified, applicable to both super- and sub-classes.

[2] Braille dataset: https://zenodo.org/records/7050094



(a)

(b)



(c)

Fig. 4. Example of neuronal response and spike encoding generated using the Mihalaş-Niebur model. The original, unchanged, data visualization is shown in (a). The spike-pattern classification results, presented in the new "Spike-Pattern Visualization" tab, are displayed with super-classes in (b) and with the complete set of behaviors in (c).

5. Known limitations and future work

The current spike-pattern classifier is trained on 1000 time-step data, necessitating slicing for longer inputs, which constrains flexibility. Addressing this limitation could significantly enhance adaptability to varying data lengths.

The classification results lack insights into the neuron's resilience to input perturbations, parameter variations, or the specific operating point. *Pignari et al.* (2024) proposed a method for examining these factors by analyzing spike-patterns within a graphical parameter space representation [7]. Adopting a similar approach –allowing exploration of spike-pattern landscapes as functions of neuron model parameters – could facilitate in-depth analysis of neuronal stability and transition criteria.

Another limitation is classification time. Reducing computational time could be achieved through dynamic batch-size adaptation for optimized GPU utilization and enhanced parallel processing.

CRediT authorship contribution statement

Vittorio Fra: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Simon F. Müller-Cleve:** Writing – review & editing, Writing – original draft, Visualization, Software, Data curation, Conceptualization. **Gianvito Urgese:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition. **Chiara Bartolozzi:** Writing – review & editing, Validation, Supervision, Resources, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Vittorio Fra reports financial support was provided by Euuropean onion. iianvito orgese reports financial support was provided by Euuropean onion. If there are other authors, they declare that they have no known competng financial interests or personal relatonships that could have appeared to infuence the work reported in this paper.

Acknowledgments

We would like to express our sincere gratitude to Elisabetta Chicca and Michele Mastella (Zernike Institute for Advanced Materials and CogniGron Center, University of Groningen) as well as Giacomo Indiveri and Nicoletta Risi (Institute of Neuroinformatics, University of Zurich and ETH Zurich) and Riccardo Pignari (Politecnico di Torino) for their invaluable discussions, insightful ideas, and unwavering motivation throughout this project. This Research is funded by the European Union - NextGenerationEU Project 3A-ITALY MICS (PE0000004, CUP E13C22001900001, Spoke 6) and the Fluently project with Grant Agreement No. 101058680. We acknowledge a contribution from the Italian National Recovery and Resilience Plan (NRRP), M4C2, funded by the European Union – NextGenerationEU (Project IR0000011, CUP B51E22000150006, "EBRAINS-Italy").

References

[1] Müller-Cleve SF, Quintana FM, Fra V, Galindo PL, Perez-Peña F, Urgese G, et al. WiN-GUI: A graphical tool for neuron-based encoding. SoftwareX 2024;27(May):101759. http://dx.doi.org/10.1016/j.softx.2024.101759, https://linkinghub.elsevier.com/retrieve/pii/S2352711024001304.

[2] Mihalaş Ş, Niebur E. A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. Neural Comput 2009;21(3):704–18. http://dx.doi.org/10.1162/neco.2008.12-07-680, URL https://direct.mit.edu/neco/article/21/3/704-718/7389.

[3] Müller-Cleve SF, Fra V, Khacef L, Pequeño-Zurro A, Klepatsch D, Forno E, et al. Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware. Front Neurosci 2022;16. http://dx.doi.org/10.3389/fnins.2022.951164, arXiv:2205.15864, URL https://www.frontiersin.org/articles/10.3389/fnins.2022.951164/full.

[4] Cramer B, Stradmann Y, Schemmel J, Zenke F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. IEEE Trans Neural Netw Learn Syst 2022;33(7):2744–57. http://dx.doi.org/10.1109/TNNLS.2020.3044364, arXiv:1910.07407, URL https://ieeexplore.ieee.org/document/9311226/.

[5] Fra V, Forno E, Pignari R, Stewart TC, Macii E, Urgese G. Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications. Neuromorphic Comput Eng 2022;2(1):014006. http://dx.doi.org/10.1088/2634-4386/ac4c38, URL https://iopscience.iop.org/article/10.1088/2634-4386/ac4c38.

[6] Forno E, Acquaviva A, Kobayashi Y, Macii E, Urgese G. A parallel hardware architecture for quantum annealing algorithm acceleration. In: 2018 IFIP/iEEE international conference on very large scale integration, vol. 2018-Octob, IEEE; 2018, p. 31–6. http://dx.doi.org/10.1109/VLSI-SoC.2018.8644777, URL https://ieeexplore.ieee.org/document/8644777/.

[7] Pignari R, Fra V, Macii E, Urgese G. Exploring spiking neuron model behaviours through the analysis of parameter space. In: International workshops of ECML PKDD 2024, Vilnius, Lithuania, September 9–13, 2024, revised selected papers. Springer, [in press].