VS2N : Interactive Dynamic Visualization and Analysis Tool for Spiking Neural Networks

Hammouda Elbez, Mohammed Kamel Benhaoua, Philippe Devienne, Pierre Boulet

▶ **To cite this version:**

Hammouda Elbez, Mohammed Kamel Benhaoua, Philippe Devienne, Pierre Boulet. VS2N : Interactive Dynamic Visualization and Analysis Tool for Spiking Neural Networks. Content-Based Multimedia Indexing, Jun 2021, Lille, France. hal-03267042

HAL Id: hal-03267042
https://hal.science/hal-03267042

Submitted on 22 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

VS2N : Interactive Dynamic Visualization and Analysis Tool for Spiking Neural Networks

1st Hammouda Elbez
Computer Science Department,
University Mustapha Stambouli of Mascara
Mascara , Algeria
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL,
F-59000 Lille, France
0000-0002-4444-1196

2nd Mohammed Kamel Benhaoua
Computer Science Department,
University Mustapha Stambouli of Mascara
Mascara, Algeria
LAPECI Laboratory, University of Oran1
Oran, Algeria
0000-0002-6145-1951

3rd Philippe Devienne
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL,
F-59000 Lille, France
0000-0002-7023-1088

4th Pierre Boulet
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL,
F-59000 Lille, France
0000-0002-0373-4478

***Abstract*—Bio-inspired computing architectures enable ultra-low power consumption and massive parallelism using neuromorphic computing, which is apt to implement Spiking Neural Networks (SNN). Such architectures are particularly suitable for energy-constrained applications. A deeper understanding of Spiking Neural Networks (SNN) behavior during training is needed to improve these architectures. This paper presents VS2N, a web-based tool for interactive visualization and analysis of SNN activity over time. This simulator-independent tool offers a way to examine, analyze and validate different hypotheses about SNN activity. We present available analysis modules and use-cases of the tool as an example.**

***Index Terms*—Spiking neural networks, neuromorphic computing, visualization, analysis, big data.**

I. Introduction

Bio-inspired technology has attracted attention lately due to the advantages that such technology offers, in particular the massive parallelism and low power consumption, which makes it suitable for energy-constrained applications, especially natural data processing applications. This technology provides neuromorphic computing by using Spiking Neural Networks (SNNs) and it is considered as one of the most promising alternatives to the Von Neumann architecture for "more-than-Moore" computing.

With the increasing amount of data and the different data-driven applications, managing multimedia data is becoming a challenge and an active research field, involving media indexing, classification or retrieval in a limited time. Recently, the use of neural networks in the field of multimedia has provided significant progress, thanks to the different applications in tasks like image classification [1] or Content-based image retrieval [2], [3].

One of the reasons why Spiking Neural Networks can be suitable for multimedia related tasks is that in contrast to classic neural networks, SNNs consider time during activity and process natural signals while being robust to noise. In [4], the author presents an SNN framework for sound classification, which has proven to perform robust sound recognition tasks and achieves promising performance. Another work [5] used Spiking Cortical Model for content-based retrieval. It produces better performance due to noise robustness and geometry invariance that it provides for features extraction and texture retrieval of images.

Due to the asynchronous nature of Spiking Neural Networks, it is crucial to understand how the network is evolving, to help improve the network performance in the different applications mentioned earlier, and easily tune the network parameters to achieve desired results. In addition, a better understanding of the Spiking Neural Networks learning behavior helps close the gap between SNNs and the classic neural networks in term of performance. In the SNN domain, many simulators that provide the possibility to experiment [6]–[10]. Most of those simulators propose basic visualization of the network activity as described in [11]. However, those visualizations are not suitable for analysis purposes since they are static and do not offer the possibility to analyze the evolution of the network parameters over time due to the enormous size of the data to collect (traces of simulation or execution). This big data makes it challenging to do the analysis manually, without a dedicated tool to process collected data. This paper presents VS2N, a web-based tool for post-mortem interactive dynamic visualization and analysis of Spiking Neural Network simulations. This tool offers a way to follow the network learning behavior over time, to move back and forth in time and various modules for analytic purposes.

The rest of this paper is composed as follows: Section II represents related works concerning visualization and neural networks. Section III presents analysis requirements. Section IV presents more technical details on VS2N and the different analysis modules. In Section V, we present use-cases to showcase the utility of VS2N. Finally, we discuss

the limitations of VS2N and future works in Section VI.

II. Related work

With the increasing interest in Neural networks from different fields, an increasing need to better understand and intercept causality in this type of network has appeared, mainly when used in critical domains. Neural Networks are widely considered as a Black-box since it is often hard to interpret some results, and one of the ways to overcome this issue is to use visual analytics. One of the first works in classic neural networks that used visualization to understand the network behavior better was [12]. The author reconstructed the features detected by the network from the last layer to the input layer to understand better how the network is reacting to input images.

In Convolutional Neural Networks (CNN), many tools exist to answer a specific question or provide a better understanding of the network behavior, especially when dealing with deep learning and deep neural networks. In [13], the authors introduce a tool for interactive features selection to understand better how predictive features are ranked across feature selection algorithms. This tool leads to essential insights when tested on a case study from the clinical research field. Other tools presented for deep neural networks (DNNs) [14], [15] help the user to better explore complex DNNs, by providing visualization approaches to convolutional neural network layers. Besides, a similarity display can reveal how each layer perceives the input in a deep neural network. In [16], a visual analysis tool is presented for recurrent neural networks (RNNs) to understand the hidden state dynamics, which leads to a better tuning of the network.

For Spiking Neural Networks (SNNs), due to the asynchronous nature and spikes for communication, analyzing large networks is challenging. In [17], they present a framework for immersive and intuitive 3D visualization of the network in virtual reality, which improves the user's abilities to explore and analyze SNNs. Another work [18] presents a 3D visualization tool of SNN by visualizing individual neurons and their connections while providing interactive control over the 3D visualization. This work focuses on the clarity of the network exploration and the implementation issues related to 3D network representation. Another multi-purpose tool is presented in [19] to visualize the network layers in 2D and 3D, using coordinated multiple views for massively parallel neurophysiological data.

If we analyze the presented tools, we can see that they have some points in common, such as being web-based tools, targeting a specific problem or question and are limited to be used with a specific simulator. We can summarize the novelty of VS2N in four points: 1. Modular nature: we consider the visualizations as modules. Anyone can add new modules for a specific analysis. 2. Simulator-independent: we can use any simulator as long as the collected data follows particular schemas, which VS2N supports. 3. Scalability: backed by the combination of Apache Spark and MongoDB for data processing, we can deploy VS2N on multi-nodes or cluster for better performance. 4. Dynamic analytics: VS2N provides the possibility to move in time with the evolution of the network, which is not possible with most of the existing tools based on analyzing static data.

In MongoDB, we store data in collections, and each collection contains documents in JSON, which are similar to rows in a relational database. A *document schema* is a JSON object that contains information about the shape, fields and type of data stored in that document. In VS2N, we need predefined schemas of the used collections to read the stored data, and simulators need to respect it to use VS2N. More information about the collections and their schemas for VS2N in https://gitlab.univ-lille.fr/bioinsp/VS2N/-/wikis.

III. Requirements analysis

For the analysis purpose of analyzing Spiking Neural Networks, we can divide the network components into three entities:

A. Input Data

Neural networks are considered data-driven. That is why the quality of the input data is critical for network performance. In real-life applications, preparing and cleaning the input data is a time-consuming operation. The analysis consists of studying and observing the input data during the training, and the network reaction, for any possible improvements.

B. Neurons

Neurons are the main components of the network. The behavior of each neuron depends on the type of the neuron and the activity in the network. The neuron analysis involves the membrane activity, the spike frequency, and the neuron behavior based on the input. By analyzing all neurons, we can check the network performance and detect any possible improvements or insights.

C. Synapses

Synapses keep the neurons connected and help transmit activity over the network. The number of synapses in a network is more than the number of neurons, making it challenging to analyze their activity. By analyzing synapses, we can learn more about the neuron's reaction to specific input and the learned features, which will be more challenging in multi-layer networks. The analysis may help eliminate unnecessary synapses while preserving good performance.

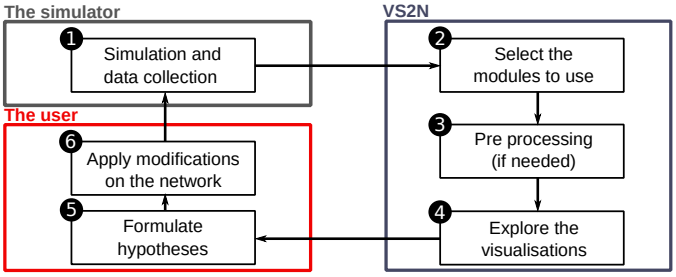

Fig. 1. The visual analysis workflow

As represented in Figure 1, to use VS2N, the user would (1) start by collecting the data from the simulation. In our case, we used the N2S3 simulator [6] and Nengo [7]. We can use any simulator as long as the collected data follows the standard schemas from VS2N. (2) select the simulation and the modules on VS2N. (3) launch the pre-processing by VS2N (if needed), using Apache Spark. (4) start exploring the visualizations and detect any patterns or phenomena. (5) formulate hypotheses based on the observations. (6) apply modifications to the network and start the simulation and data collection again.

IV. VS2N

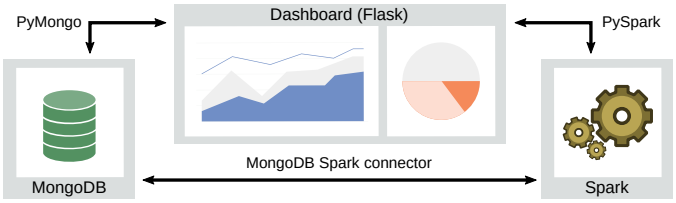

Fig. 2. VS2N components and used libraries for communication

VS2N is a web-based tool based on Flask [20], a micro web framework written in Python. The backend is composed of two main parts: MongoDB[1], for storing data collected from the simulation and Apache Spark[2], for any required pre-processing on the data. Due to the nature of MongoDB and Apache Spark. This combination makes it possible to scale in terms of computation power and deploy on the nodes of a distributed cluster (see Figure 2). VS2N uses Dash library[3] to create web interfaces and interactive visualizations using Python.

In this section, we discuss the different analysis modules in VS2N, the used visualizations, and the purpose of each module.

A. General Analytics Module

This module represents an overview of the network performance and general information, like the number of neurons and layers, network accuracy, and the used dataset. This module is the first one used with every analysis (see Figure 3). It is composed of three main parts:

1) **General information:** represents static information on the network accuracy, used topology, and dataset.
2) **Network activity:** a general visualization of the network activity, such as spikes, neurons potential, synapses update, and loss update during training. This part is helpful to observe any repetitive pattern in the activity, the learning process of the network, and any correlation between the different graphs.
3) **Dataset overview:** represents insights on the actual input of the network at that period. This visualization represents the number of each input (grouped by labels if it is a labeled dataset, otherwise this visualization is hidden), and it does not depend on the learning type (supervised or unsupervised) but only on the type of dataset (labeled or not).

[1]www.mongodb.com
[2]www.spark.apache.org
[3]www.plotly.com/dash

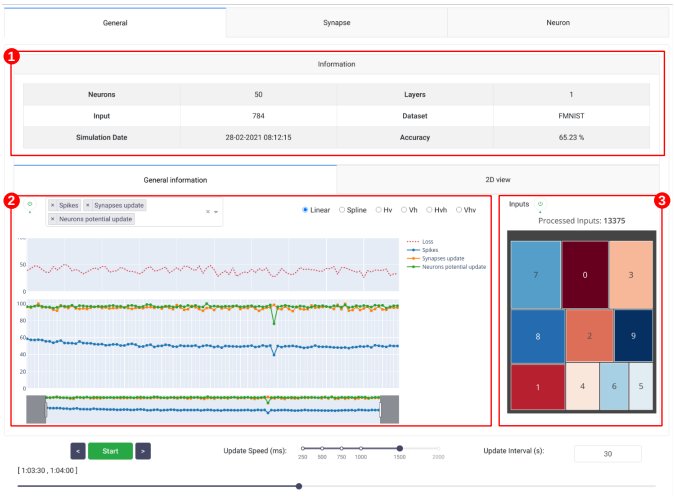

Fig. 3. VS2N: General Analytics Module

B. Neuron Analytics Module

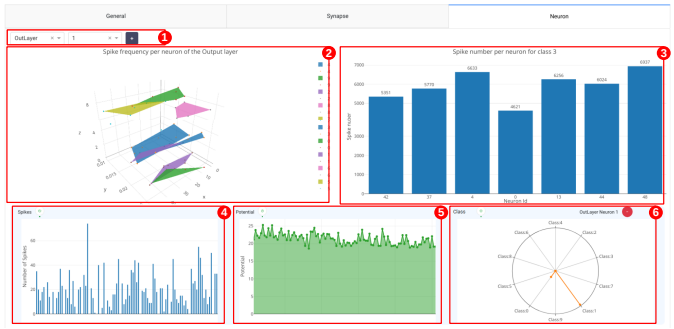

Fig. 4. VS2N: Neuron Analytics Module

This module contains a group of interactive visualizations to observe the activity of each neuron individually or grouped (see Figure 4). The different components of this module are:

1) **Layer and neuron selector:** when the user selects a neuron, upon this selection, more visualizations regarding this element are displayed (4, 5, and 6).
2) **3D spike frequency:** this component provides a 3D representation of the spike frequency per neuron in the output layer (neuron id on the X-axis, spike frequency on the Y-axis, and Z-axis for the detected class). This visualization gives the user insight into the spike frequency of the neurons compared to the detected class.
3) **Neuron spikes per class:** this component is visible only when a neuron from a class is selected in the 3D spike frequency component. This component provides information on the number of spikes per neuron in the selected class. This component is complementary to the

previous one, which provides more information for a better analysis of the neuron activity in the same class.

Once the user selects one or more neurons, VS2N adds three new visualizations to the screen. Those visualizations are:

4) **Neuron spikes activity:** which represents the spiking activity of the selected neuron.
5) **Neuron potential activity:** this represents the membrane potential activity of the selected neuron.
6) **Neuron class activity:** it represents the input class that caused the selected neuron to spike. It is visible only if the dataset is labeled.

Using the presented visualizations, the user can analyze the behavior of the selected neuron with the help of spikes and potential activity while considering the input that caused the neuron to spike.

C. Synapse Analytics Module

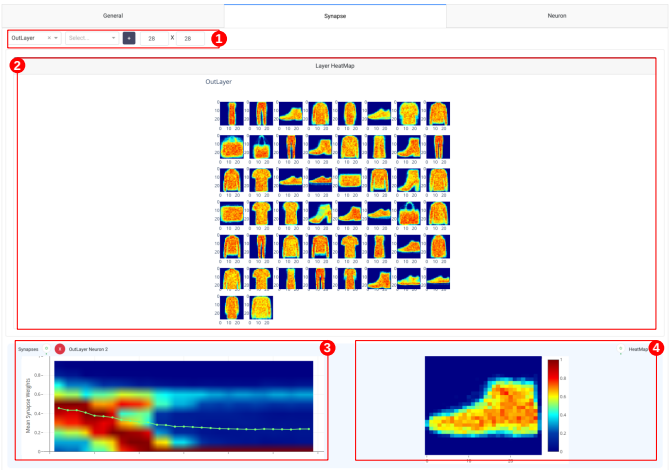

Fig. 5. VS2N: Synapse Analytics Module

This module contains a group of interactive visualizations to observe the synaptic activity of each neuron (see Figure 5). The Components of this module are:

1) **Layer and neuron selector:** when the user selects a neuron, upon this selection, more visualizations regarding this element are displayed (3 and 4).
2) **Layer heatmap:** it represents an overview of the selected layer heatmap at the end of the training. This visualization helps the user to get a clear overview of the features detected by the selected layer, which is helpful to evaluate the learning process.

Once the user selects one or more neurons, VS2N adds two other components, representing information about the activity of the selected neuron. Those components are:

3) **Synapses weight activity:** this represents the mean value of the selected neuron weights and a vertical heatmap of the weights distribution activity during the recorded period.
4) **Neuron heatmap:** this component is complementary to the previous visualization. It represents the synapses heatmap of the selected neuron. The main difference between this representation and (2) is that this representation gets updated over time while the layer heatmap represents the final values of synaptic weights.

These visualizations provide insight into how the synapses react to input data. Any modification on the detected class by the neuron can be spotted by a slight variation in the mean synapse weights and the heatmap.

V. Applications

In this section, we present two use-cases, where we use VS2N to observe an activity, answer a question or validate a hypothesis. For the simulation, we used the N2S3 simulator, an open-source, scalable spiking neuromorphic hardware simulator [6], and the Nengo simulator [7], which offers the possibility to use neuromorphic hardware such as Loihi and SpiNNaker. MNIST dataset [21] was used for testing. It contains handwritten digits (60k for training, 10k for testing).

A. Use-Case: The MNIST last 10k effect

In this use case, we will try to validate one observed phenomenon during simulation using a single layer network. This phenomenon consists of an accuracy drop in the last 10k of the MNIST dataset during training. This accuracy drop is seen when using small or medium networks (with less than 1000 neurons). However, it does not appear in large networks, which leads to the assumption that in the last 10k input, new variations of handwritten digits are introduced to the network. As a result, by training the same network while inverting the MNIST dataset, this drop disappears, as seen in Figure 6, but appears in the first 10k.

For this use case, we use Nengo for simulation.

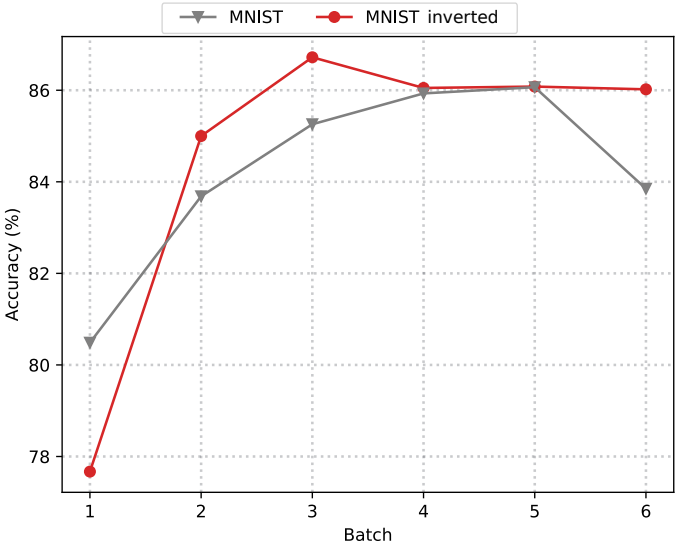

Fig. 6. Network accuracy using 900 neurons with MNIST (1 batch = 10k)

Using VS2N, we will monitor the network activity during training and the last 10k using the general analytics module. The goal is to validate the existence of this phenomenon.

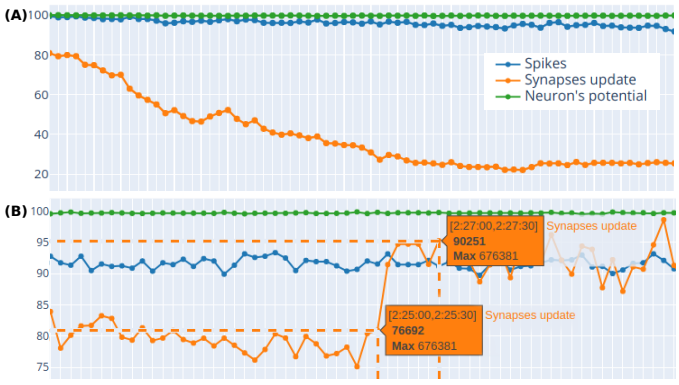

Fig. 7. Network activity, (A) during the first 15k input, (B) during the last 10k

As we can see in Figure 7(A), at the start of the training, the number of spikes and neurons potential update is stable, while synaptic weights update is decreasing and becomes stable after a couple of inputs, which is expected due to training and the nature of MNIST dataset. After 50k of input, we can see an interesting pattern at the start of the last 10k (Figure 7(B)), with a clear increase in synaptic weights update (more than 10%) that lasts until the end of the training. This increase in synaptic weights update means that the patterns learned by the neurons are changing, which will affect the network performance, as reported in Figure 6.

B. Use-Case: Network Compression

One technique to compress the network is by pruning synapses that are considered not critical to the network. The criterion of selection is usually based on a defined threshold. If the synapse weight value is below the threshold, the synapse is considered not critical and removed. The threshold selection is usually tuned experimentally, and it is interesting to observe the effect of the prune operation on the network when executed during training.

For this use case, we use N2S3 to simulate a single-layer network and apply prune operation (threshold=0.2) after training using half of the MNIST dataset. Using VS2N, we will monitor the general activity of the network once pruned and the effect on randomly selected neurons.

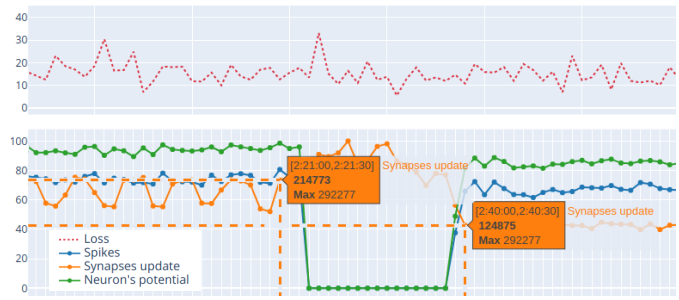

Fig. 8. Network activity and loss graph after 30k input (during the prune operation)

In Figure 8, we can see the activity of the network before and after the prune operation. The spikes and neurons' potential activity had a slight decrease in value after removing more than 50% of the synapses, which is expected since only the weak synapses were removed due to the MNIST dataset's nature that contains centered handwritten digits. We can see a drop in values and a more regular graph after pruning for the activity of the synaptic weights. Although the removed synapses are weak, they are still updated during training, as shown in the graph. We can see a slight increase just after pruning for the loss graph, but it goes back to normal again. We can see in Figure 8 that after the prune operation, there was a period of silence in the network. This is due to the pruning process in the simulator, which affects the simulation time.

In Figure 9, we can see the synaptic activity of two neurons using the synapse analytics module. After 15k of input (Figure 9(A)), we can see the mean value of synapse weights is close for both neurons. However, the distribution of the synaptic weights over time (left) is different since the two neurons are learning different classes, as we can see in the heatmap (right). In Figure 9(B), we can see the effect of applying the prune operation on the two visualizations. The mean synaptic weights increased since weak synapses were removed, and we can see a change in the distribution of the synaptic weights (left). The heatmap (right) shows that the selected threshold (0.2) does not affect the detected class, and the network will produce good performances compared to the non-pruned version.

VI. Conclusion

The ability to explain the results of a neural network plays an essential role toward a better exploration of this technology, especially in critical applications where every decision should be justified.

In this work, we presented VS2N, a simulator-independent tool for dynamic visual analysis for Spiking Neural Networks. VS2N offers the possibility to analyze the network activity interactively, with the possibility to add new modules. We presented two use cases with data collected from two different simulators.

This version mainly supports the analysis of shallow networks. Due to the complexity of the multilayer networks, more work is needed to add modules that suit this type of networks, like features reconstruction and 3D visualization of the activity between the hidden layers. The pre-processing phase at the first time (using Apache Spark) takes time due to the massive amount of data, which we can reduce by deploying VS2N on the nodes of a distributed cluster. We will address all those limitations in the future.

Finally, it is worth mentioning that VS2N can be used to analyze the activity during the simulation without the need to wait for the simulation to finish, which can be useful when simulating large networks that can take much time. However, only the general analytics module will be available in this case since the other modules require pre-processing, which is not possible to do while simulating.

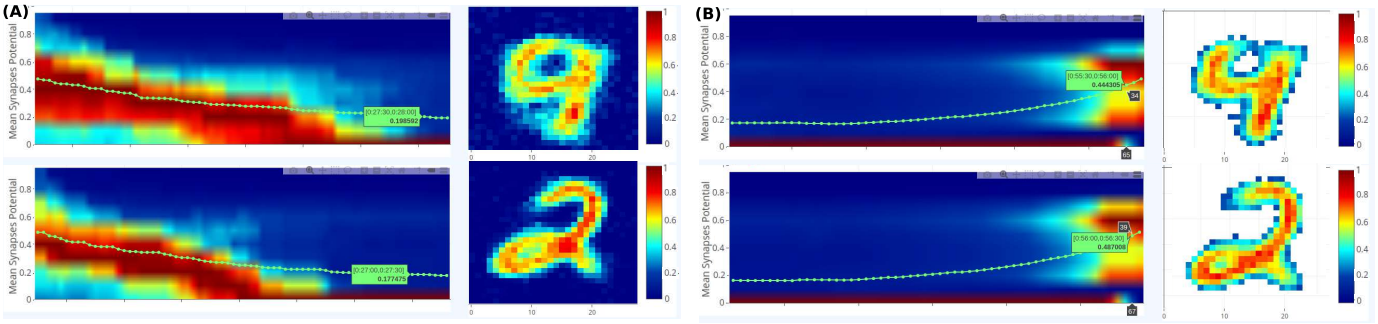

Fig. 9. Synapses activity of two randomly selected neurons, (A) during the first 15k input, (B) after 30k input

ACKNOWLEDGMENTS

This work was supported in part by IRCICA (Univ. Lille, CNRS, USR 3380 – IRCICA, F-59000 Lille, France) under the Bioinspired Project, PROFAS B+ 2019 and PRIMA program under grant agreement No 1821, project WATERMED4.0. The PRIMA programme is supported by the European Union.

REFERENCES

[1] P. Falez, P. Tirilly, I. Marius Bilasco, P. Devienne, and P. Boulet, "Multi-layered spiking neural network with target timestamp threshold adaptation and stdp," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[2] F. Sabahi, M. Omair Ahmad, and M. N. S. Swamy, "An unsupervised learning based method for content-based image retrieval using hopfield neural network," in *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, 2016, pp. 1–5.

[3] A. Sezavar, H. Farsi, and S. Mohamadzadeh, "Content-based image retrieval by combining convolutional neural networks and sparse representation," *Multimedia Tools and Applications*, vol. 78, no. 15, pp. 20 895–20 912, Aug 2019. [Online]. Available: https://doi.org/10.1007/s11042-019-7321-1

[4] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in Neuroscience*, vol. 12, p. 836, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00836

[5] R. Yang, C. Lyu, Y. Liu, W. Zhou, C. Chen, X. Jiang, P. Li, H. Chen, R. Xu, and Y. Wang, "Spiking cortical model for geometry invariant and antinoise texture retrieval," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2017, pp. 645–650.

[6] P. Falez, P. Devienne, P. Tirilly, M. Bilasco, C. LOYEZ, I. Sourikopoulos, and P. Boulet, "Flexible Simulation for Neuromorphic Circuit Design: Motion Detection Case Study," in *Conférence d'informatique en Parallélisme, Architecture et Système (ComPAS)*, Sophia Antipolis, France, Jun. 2017.

[7] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, p. 48, 2014.

[8] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, Apr. 2007.

[9] M. L. Hines and N. T. Carnevale, "The neuron simulation environment," *Neural Computation*, vol. 9, no. 6, pp. 1179–1209, 1997.

[10] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, p. e47314, Aug. 2019.

[11] H. Elbez, K. Benhaoua, P. Devienne, and P. Boulet, "Visualization Techniques in SNN Simulators," in *3rd International Conference on Multimedia Information Processing, CITIM'2018*, Mascara, Algeria, Oct. 2018.

[12] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520–1528.

[13] J. Krause, A. Perer, and E. Bertini, "INFUSE: interactive feature selection for predictive modeling of high dimensional data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 1614–1623, Dec 2014.

[14] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 88–97, 2018.

[15] M. Zurowietz and T. W. Nattkemper, "An interactive visualization for feature localization in deep neural networks," *Frontiers in Artificial Intelligence*, vol. 3, p. 49, 2020.

[16] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 667–676, 2018.

[17] S. Marks, "Immersive visualisation of 3-dimensional spiking neural networks," *Evolving Systems*, vol. 8, no. 3, pp. 193–201, Sep 2017.

[18] A. Kasiński, J. Pawłowski, and F. Ponulak, "'snn3dviewer' - 3d visualization tool for spiking neural network analysis," in *Computer Vision and Graphics*, L. Bolc, J. L. Kulikowski, and K. Wojciechowski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 469–476.

[19] J. Senk, C. Carde, E. Hagen, T. W. Kuhlen, M. Diesmann, and B. Weyers, "Viola—a multi-purpose and web-based visualization tool for neuronal-network simulation output," *Frontiers in Neuroinformatics*, vol. 12, p. 75, 2018.

[20] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 1st ed. O'Reilly Media, Inc., 2014.

[21] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2010.