Visualization Techniques in SNN Simulators

Hammouda Elbez, Kamel Benhaoua, Philippe Devienne, Pierre Boulet

Visualization Techniques in SNN Simulators

Hammouda Elbez, Kamel Benhaoua
University of Oran1 Ahmed Ben Bella -LAPECI- Mustapha Stambouli University
Mascara, Algeria
{Elbez.Hammouda, k.Benhaoua}@univ-mascara.dz

Philippe Devienne, Pierre Boulet
Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille
F-59000 Lille, France
{Philippe.Devienne, Pierre.Boulet}@univ-lille.fr

***Abstract*—Neural networks are one of the most well-known artificial intelligence technique. These networks have known a huge evolution since their first proposal, represented by the three known generations. On the other hand, neuromorphic architectures are a very promising way to overcome the limitation of the Von Neumann architecture and the end of Moore's law. Neuromorphic architectures can lead to a huge energy consumption decrease mostly due to the colocation of computation and storage.**

These architectures implement spiking neural networks (SNNs), the third generation of artificial neural networks that model very finely biological neural networks. One of the main problematics that prevents us from optimizing and getting the best performance of SNNs and as a result the neuromorphic architectures development and production, is the lack of a clear and complete understanding of their behavior, especially what makes learning efficient. One of the approaches to answer that is analyzing by visualization of simulation traces of such networks and architectures. In this paper, we propose a comparison of the visualization techniques proposed by SNN simulators for analysis purposes.

***Index Terms*—Neural Network, Data Science, Visualization, Simulation, Analysis, Neuromorphic Architectures, SNN.**

I. Introduction

In the last years, neural networks have had a big impact in many fields like image processing and decision making systems. Being able to understand and to model such networks was a key to achieve the remarkable performances which can be observed in the second generation of neural networks (Convolutional Neural Networks, ConvNets). These performances come at the price of a huge energy consumption, especially during learning. On the contrary, the brain is very power efficient, thus researchers have tried to design artificial neural networks whose behavior is more precisely modeled after the biological neural networks. These Spiking Neural Networks (SNNs) are indeed much more power efficient than ConvNets and are suitable to electronic implementation, the so called neuromorphic architectures.

Before any implementation, the need of passing by simulation is very important and for this type of networks, many simulators have been created and used over the last years to get and test the suitable network configuration (or to simulate biological SNNs). Those simulators offer also a variety of visualization techniques that can lead to very interesting observations and improvements in network performance, the combination of these visualization techniques and the best choice to use, can have an important influence on the user experience, the information that can be extracted from it and the ability to improve the performance of neuromorphic architectures.

The aim of this work is to present the visualization techniques offered by the known SNN simulators and discuss their ability to produce visualizations suitable to analyze the behaviour of the simulated networks in order to improve neuromorphic architectures. This paper is structured as follows. After a briefing introduction to neural networks and SNNs, we present the state-of-the art of SNN simulators. We then compare these simulators from the technical and visualization points of view and discuss our findings.

II. Neural Networks

Neural networks are an architecture inspired from our brain, it contains a large number of simple interconnected units which simulate the neurons. Each unit (neuron) has multiple inputs and outputs neurons. The first generation of neural network was able to solve only linear problems, it is represented by the formal mathematical models like McCulloch and Pitts neurons [1], Perceptron, Hopfield network and Boltzmann machine. In the second generation where multi layer neural networks have been introduced, it was able to solve non-linear problems by the use of continuous activation function, for example the MultiLayer Perceptron (MLP).

Being inspired from our brain, we can find the learning term and this phenomenon exists also in this type of networks. In the life cycle of every neural network, there are mainly two phases which are: the learning (tuning) phase and the use (production) phase. The learning phase consists of tuning the network parameters using test dataset in order to get the best configuration with the lowest error rate and the use phase is the actual use of this network with real data. The learning in neural networks can be classified in more then one type, which are: supervised learning, unsupervised learning, half supervised learning and learning by strengthening. The supervised learning represents a learning way based on a tested dataset that are labeled, which means that every entry is composed from two information, the actual data and the label that describes it. The unsupervised learning, this type does not provide a labeled learning dataset and learns by itself to classify the inputs into categories, this type is very interesting in real life scenarios because it is very consuming

to make a labeled dataset and having a network working with this type of learning increases its ability to adapt for many use cases, but may take more time in learning process compared to the first type of learning in order to achieve a good performance. Half supervised learning, this type exists because having labeled dataset is often a very consuming task in time and resources, that is why having a hybrid dataset (labeled and not labeled) can reduce negative points known in supervised and unsupervised types of learning (such as time and resource consumption). Learning by strengthening, which is a method used while the network is in the use phase, by keeping the tuning of this network up and running to increase its performance using the information that is actively processing.

A. Neuron

A neuron is the principal component of a neural network, it has many input and output neurons. The functionality of a single neuron is simple and useless alone, but the global activity of the million connected neurons is what makes it powerful. Scientists in biology, electronics and computer science field have tried to understand this component due to its advantages and abilities. From the biological point of view. Like it is shown in Fig. 1, a neuron is composed of: The membrane, which is considered as the core of the neuron where ions activity happens. Synapse, that is the interaction space between two neurons to transfer information, this space uses liquid channels and ions for transportation. Dendrite, which represent the element that is responsible for transmitting the information after being received by the synapse into the membrane. Axone, that is the output support, used by the neuron to send out the information, it can vary in length from some millimeters to many meters.

In the electronics field, scientists have tried to model such a component for use in hardware due to the advantages it offers, this type of architectures which are called neuromorphic are considered actually as an active research field in particular with the appearance of Memristor [2], that was produced for the first time by HP [3]. In the computer science field, many models have been proposed in order to implement such components, the main elements of a neuron have got their equivalent in computer science field that provides the same role, such as synaptic weights which represent the biology synapse, the activation function which represents the membrane and the output elements which represent the axone.

B. SNN

Spiking neural networks are considered as the main model in the neuromorphic architectures which are a very promising approach to replace the actual Von Neumann architecture because of the way memory and information are processed in this kind of networks [4], this type of networks is the closest one to what we have in our brain due to the fact that SNN use spikes for communication, they offer a real parallelism, fault tolerance and a low energy consumption thats what make it an attractive alternative. SNN has the ability

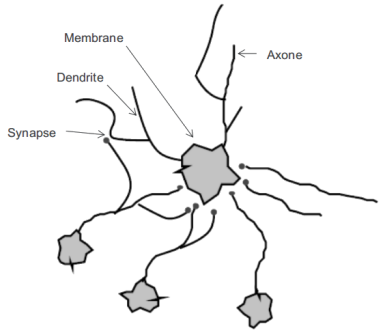

Fig. 1. Neuron.

to process a big amount of data with a small number of spikes, they emulate data processing, plasticity and learning in our brain. Spiking neural network is a network composed of a big number of neurons connected by synapses (millions of neurons and more synapses), the communication between those neurons are made by spikes that are considered as the language of communication, the information is coded inside a single spike and many neural information encoding have been proposed based on what is known in biology [5]. This network has known the apparition of time notion, that is considered as a very important factor in performance and the information transfer process, some scientists have even proposed that time can actually hold information in this type of network and considered as information coding technique [6]. One of the reasons why this type of network has got attention lately, is the discovery of the memristor, this element is capable of emulating the functionality of the synapse and helps implementing such networks in hardware which was not easy to implement before. [7]

III. SNN Simulators

Simulation is an important step in every experiment, like in SNN before going to implementation of any kind of architectures, we have to pass by simulation in order to conduct the required tests. Many simulators have been created in order to help researchers to test their hypotheses. In this paper, we are interested in simulators that offer visualization of this type of network, which are: Neuron, Brian, Nengo, Neuronify, Simbrain, N2S3 and NEST.

SNN are essentially defined by standard differential equations, but because of the discontinuities caused by the spikes, designing an efficient simulation of spiking neural networks is a non-trivial problem. There are two families of simulation algorithms: event-based simulators and clock-based ones. Synchronous or clock-driven simulation simultaneously updates all the neurons at every tick of a clock, and is easier to code, especially on GPUs, for getting an efficient execution of data-parallel learning algorithms. Event-driven simulation behaves more like hardware, in which conceptually concurrent components are activated by incoming signals (or events) [8].

Event-driven execution is particularly suitable for untethered devices such as neurons and synapses, since the nodes can be

put into a sleep mode to preserve energy when no interesting event is happening. Energy-aware simulation needs information about active hardware units and event counters to establish the energy usage of each spike and each component of the neural network. Furthermore, as the learning mechanisms of spiking neural networks are based on the timings of spikes, the choice of the clock period for a clock-based simulation may lead either to imprecision or to a higher computational cost.

There is also a fundamental gap between this event-driven execution model and the clock-based one: the first one is independent of the hardware architecture of computers on which it is running. So, event-driven simulators can naturally run on a grid of computers, with the caveat of synchronization issues in the management of event timings.

A. NEURON

NEURON is one of the oldest simulators. It was developed in 1997 by Michael Hines, John W. Moore and Ted Carnevale at Yale and Duke [9]. It offers the possibility to model individual or network of neurons. The primary scripting language of NEURON is based on HOC (High Order Calculator) programming language [10] but a Python interface is also available and can be used, with the possibility of loading programs from a file or written in a shell. It supports parallelization using the MPI protocol [11].

Visualization techniques offered by NEURON can be resumed in a variety of graphs as shown in Fig.2, such as membrane potential graph and dendrite voltage graph.

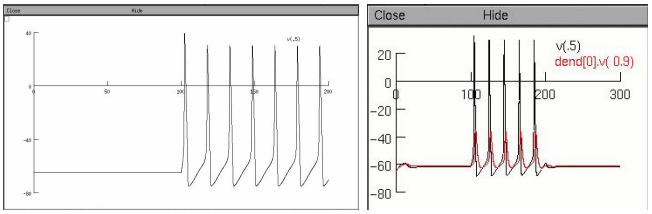

Fig. 2. Visualization example in NEURON

B. Brian

Brian was developed by Marcel Stimberg, Dan Goodman and Romain Brette, this tool was made in order to make the coding of spiking neural networks fast, easy to use and flexible. Brian was written in an interpretative language, but is still effective in many situations thanks to vectorized algorithms [12], but Brian is not made for very big simulations that need important resources or for simulating detailed biophysical models. [13]

Brain offers many possibilities of visualization some are shown in Fig.3, it has the possibility to automatically change the visualization depending on the size of the network, those visualization techniques include but not only membrane potential graph, spikes plot, firing rate graph and synapses connections representation.

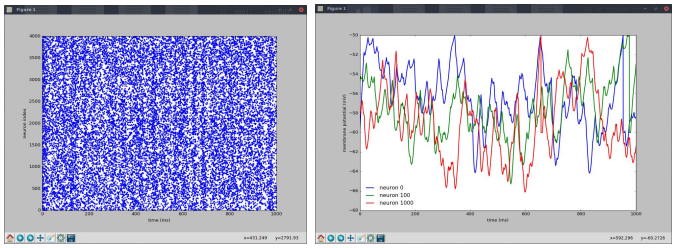

Fig. 3. Visualization example in Brian

C. Nengo

Nengo is a python library used to create and simulate very large scale neural networks, it can create spiking neural simulations and other sophisticate types in a few lines of code. [14] Nengo offers the possibility to define neuron type and learning rules, create and execute deep neural network and simulate on hardware devices such as Spinnaker [15]. Nengo offers many types of backend that can be used for simulation, which are Nengo, Nengo_ocl, Nengo_mpi, Nengo_distilled and for the hardware simulators Nengo offers two types which are Nengo_brainstorms, and Nengo_spinnaker.

Nengo is composed of a server written in python connected to Nengo core and an interactive interface for users to manipulate which is based on web technologies like HTML, D3.js and jQuery.

When it comes to user experience, Nengo provides a unique experience for users to interact with the code directly on the interface and see the changes in real time with an acceptable degree of interactivity. Nengo offers a variety of visualization techniques that can be used to follow the network activity Fig. 4, like membrane potential graph, spikes plot and activation patterns.

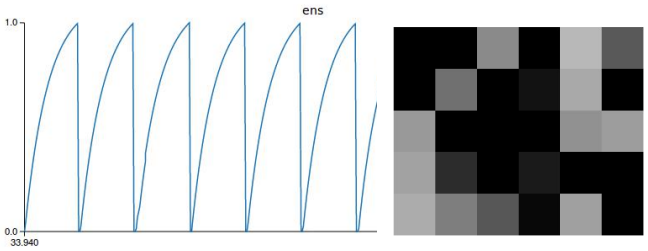

Fig. 4. Visualization example in Nengo

Nengo was used to build Spaun, the worlds largest functional brain model. Spaun currently contains 6.6 million neurons and over 10 billion synapses.

D. Neuronify

Neuronify is an educative tool created to simulate neurons and neural network behavior, it can be used to combine neurons with different connections and see how changes in individual neurons can lead to behavior change of big networks, it is developed in C++ and QML using the cross-platform application framework Qt by Ovilab [16], which is a

group of developers dedicated to creating tools for scientific and education field at the University of Oslo. [17]

In Neuronify, exploring and creating neural networks is made easy by drag and drop of the elements on the screen. It is available to use on desktop or mobile devices.

Neuronify offers a nice interface and interactivity for users and for the network visualization, Neuronify has three techniques to offer which are Spike detector plot, firing rate plot and the membrane potential graph, examples are represented in Fig. 5.

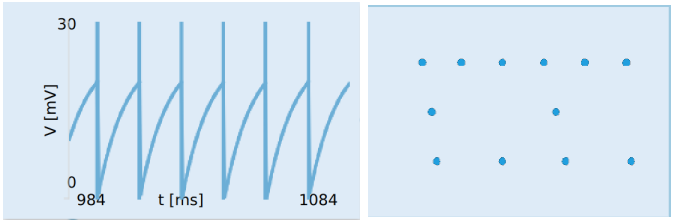

Fig. 5. Visualization example in Neuronify

E. Simbrain

Simbrain is a tool for constructing artificial neural networks, written in Java and is under GNU license, this simulator concerns more scientists in biology or medical field and this can be observed by the big number of biological representations and networks. Simbrain is created by the philosophy of making things easier for users, it comes with a variety of examples and types of networks with description, in an organized way and an easy interface to interact and work with. [18]

For the visualization techniques, Simbrain is well documented and due to the general nature of this simulator, it offers many techniques for the user to choose from as shown in Fig. 6, like Spikes plot, Spike visualizer, membrane potential variation graph and the possibility to combine more than one technique.

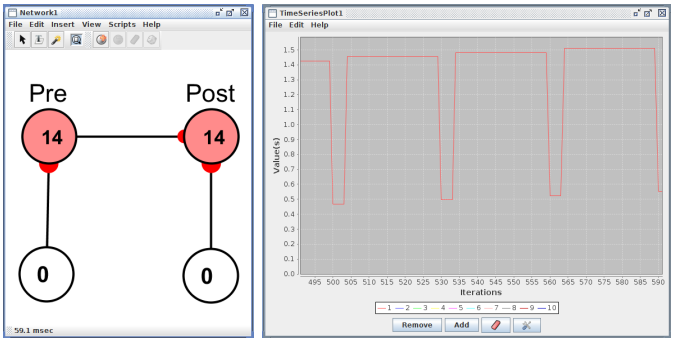

Fig. 6. Visualization example in Simbrain

F. N2S3

N2S3 (Neural Network Scalable Spiking Simulator) is a simulator created to simulate neuro-inspired hardware accelerators, it is written in Scala and Akka which give it all the scalability and portability features that Scala offers. N2S3 is considered as an event-driven simulator, which is based on exchanging messages between concurrent actors to mimic the spikes exchange between neurons [19]. Being flexible, extensible and scalable make the integration of new models or tools easy. [20]

N2S3 has been developed from the ground up for extensibility, allowing to model various kinds of neuron and synapse models, various network topologies, various learning procedures, various reporting facilities, and to be user friendly with a domain specific language to express the experiments the user wants to simulate. It is available online as open source software. It comes as a library with some classic experiments ready to use, such as handwritten digit recognition on the MNIST dataset [21] and the highway vehicle counting experiment. [22]

N2S3 offers the possibility for users to observe simulation outputs through network observers, which can vary from textual loggers to dynamic visualizations, it offers also a synaptic weight evolution visualizer, spike activity map and the visualization of actual input data processed. Fig.7.

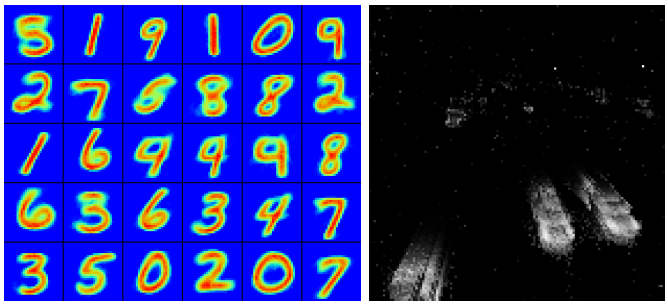

Fig. 7. Visualization example in N2S3

G. NEST

NEST is an SNN simulator coordinated by the NEST Initiative, it has a focus on the network dynamics, size and structure rather than the individual neurons. It is considered as one of the most attractive tools to use for SNN because of its ability to work with any size of network. [23] NEST offers two ways for creating simulations. First, by using it as a Python library (PyNEST), which offers many commands to access NESTs simulation kernel and conduct a simulation. Secondly, by using the stand alone application (NEST). Having the simulation kernel written in C++ provides NEST with the speed and the possibility to have optimized simulations, the used simulation language is SLI [24] which is considered as a stack oriented language.

NEST does not offer a visual interface for the user to create, edit and manage the simulation, its all done by either working with the command line offered by this tool or using NEST as a Python library.

NEST offers the possibility to view the network in text form to verify if the network was built correctly, with the help of Python packages like Matplotlib [25] and NumPy [26], it offers a variety of visualization techniques as shown in Fig.8, like Membrane potential representation, Synaptic weights representation and neurons spikes plot. Some contributions have

been made in order to create tools based on NEST simulations to conduct a visualization, like VisNEST [27] which was able to visualize a network with 20 million neurons and many more synapses in medical field.

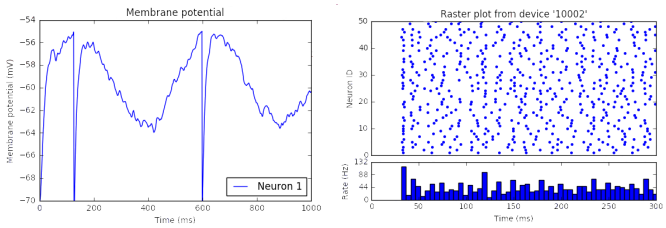

Fig. 8. Visualization example in NEST

IV. Comparison and Discussion

A. Comparison

In the data science field, many visualization techniques and models have been proposed, especially with the rise of Big Data and the need for a way to analyze such a big quantity of data, in order to give the user the best performance and experience exploring them. A good and useful visualization may lead to highlight potential models and in our case network and architecture improvement, so having the possibility to judge and compare a visualization technique is very important.

From a technical point of view, the simulators presented have interesting characteristics like the used technology and support for large scale simulations and more, presented in Table I.

From the visualization point of view, we will use seven criteria for comparison, those criteria have been chosen by combining the ones presented by Freitas [28] and those by Stephen Few which is recognized as a world leader in the field of data visualization [29] and another important criterion in our case which is the interactivity level. The criteria are divided into three categories: Informative, which contain the criteria that represents the degree to which a visualization is informative. Emotive, that represents criteria concerning design and conception and finally the Interactivity.

The first category called "Informative", contains five criteria, which are Usefulness, whether this visualization is providing a useful information or not. Completeness, if the visualization contains all the required components to be easy to understand and transmit the information or not. Perceptibility, whether it is clear and easy to understand it or not. Truthfulness, represents the degree of accuracy and validity of the visualization. Finally, Intuitiveness represents the degree of familiarity of this visualization technique for the user. The second category, called "Emotive", contains one criterion chosen for our case, Aesthetics, that concerns the design part and quality of the visualization technique. For the last criterion, it is the Interactivity level of this technique.

B. Discussion

From Table I, we can see that a variety of programming languages have been used in the development process, while Python is the most used between them due to its nature as a scripting language and the huge libraries it offers for visualization. Another thing to notice, is the parallelism and large scale simulations which are not supported by all the simulators and this may affect their performance and limit their usability while working with big networks and big data.

From the comparison of provided visualizations of the SNN simulators represented in Table II, we can see that all of those simulators offer a good level of usefulness and perceptibility, by providing a useful and easy to understand visualization. For the completeness criterion, we can see that NEURON and Neuronify does not provide a visualization with the necessary components in order to be able to get and extract useful information, the reason for that can be the old techniques provided by one of the oldest simulators which is NEURON. For Neuronify, it is the fact that this is a simple tool for beginners to discover this kind of network and is not made for scientific analysis. For truthfulness and intuitiveness, we can see that all the simulators got an easy and clear visualization techniques that are used and known by almost everyone. Almost all simulators had good score when in comes to aesthetics, except NEURON which lacks the artistic and design part in the offered techniques. Finally, for the interactivity, the basic requirements, alike zooming and moving are included in the most of the simulators except N2S3 and NEST, which is due to the fact that these two simulators do not provide visualization for the analysis purpose at the required level and for this reason tools are being developed.

All the presented simulators do very well with small networks, but when the network gets larger, not all of the simulators can perform well. The visualization techniques used for simple networks become less useful and compatible for a large scale analysis. We claim the neuromorphic computing research community needs multi-scale interactive visualization tools to better understand the learning processes in large SNNs.

V. Conclusion

In this work, we presented a brief introduction about neuromorphic architectures and spiking neural networks, where we discussed the importance of this kind of architectures and the challenges that exist and that can be resolved via a good analysis and visualization of the network activity. We then presented the state-of-the-art SNN simulators and the visualization techniques they offer. Finally, we proposed comparison criteria to analyze the visualization offered by those simulators and revealed the fact that for the purpose of improving neuromorphic architectures, we need high level interactive visualization tools designed for analysis taking in consideration multi-scale visualization and analysis for large networks.

Acknowledgments

This work was supported in part by IRCICA (Univ. Lille, CNRS, USR 3380 – IRCICA, F-59000 Lille, France) under the Bioinspired Project.

TABLE I
TECHNICAL REVIEW

	Technology	Parallelism	Type of simulation	Large scale support	GUI	Commandline
NEURON	C,C++,python	+	Event-driven	+	+	+
Brian	Python,NeuroML,PyNN	+	Clock-driven	-	-	+
Nengo	Python,Numpy,HTML	+	Clock-driven	+	+	+
Neuronify	C++, Qt	-	Clock-driven	-	+	-
Simbrain	Java	-	Clock-driven	-	+	-
N2S3	Scala,Akka	-	Event-driven	+	-	+
NEST	C++,Python,SLI	+	Event-driven and Clock-driven	+	-	+

TABLE II
VISUALIZATION COMPARISON

	Usefulness	Completeness	Perceptibility	Truthfulness	Intuitiveness	Aesthetics	Interactivity
NEURON	++	-	++	+	++	-	+
Brian	++	++	++	+	++	+	+
Nengo	++	+	++	++	++	++	++
Neuronify	+	-	++	+	++	++	++
Simbrain	++	++	++	++	+	+	+
N2S3	++	+	++	++	++	+	-
NEST	++	+	++	++	++	+	-

Good = ++, Medium = +, Bad = -

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[2] L. Chua, "Memristor-The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, pp. 507–519, Sep. 1971.

[3] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[4] G. Indiveri and S. C. Liu, "Memory and Information Processing in Neuromorphic Systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015.

[5] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6, pp. 715–725, Jul. 2001.

[6] R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in Neurosciences*, vol. 28, no. 1, pp. 1–4, Jan. 2005.

[7] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010, 01631.

[8] R. Brette *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *Journal of Computational Neuroscience*, vol. 23, no. 3, pp. 349–398, Dec. 2007.

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

[10] "Hoc - A User Extendable Interactive Language." [Online]. Available: https://www.neuron.yale.edu/neuron/static/docs/refman/hoc.html

[11] "Message Passing Interface," Jun. 2018, page Version ID: 847227383. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Message_Passing_Interface&oldid=847227383

[12] R. Brette and D. F. M. Goodman, "Vectorized algorithms for spiking neural network simulation," *Neural Computation*, vol. 23, no. 6, pp. 1503–1535, Jun. 2011.

[13] "The Brian spiking neural network simulator." [Online]. Available: http://briansimulator.org/

[14] "The Nengo neural simulator Nengo 2018-05-24 documentation." [Online]. Available: https://www.nengo.ai/

[15] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[16] "Ovilab." [Online]. Available: http://ovilab.net/neuronify/

[17] S. Dragly *et al.*, "Neuronify: An Educational Simulator for Neural Circuits," *eNeuro*, vol. 4, no. 2, pp. ENEURO.0022–17.2017, Mar. 2017.

[18] "Simbrain." [Online]. Available: http://simbrain.net/

[19] D. Wyatt, *Akka Concurrency*. USA: Artima Incorporation, 2013.

[20] P. Boulet *et al.*, "N2s3, an Open-Source Scalable Spiking Neuromorphic Hardware Simulator," Universit de Lille 1, Sciences et Technologies ; CRIStAL UMR 9189, report, Jan. 2017, 00001. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01432133/document

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[22] O. Bichler *et al.*, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks*, vol. 32, pp. 339–348, 2012, 00073.

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

[24] "Programming in sli - nest simulator." [Online]. Available: http://www.nest-simulator.org/programming-in-sli/

[25] J. D. Hunter, "Matplotlib: A 2d Graphics Environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, May 2007.

[26] T. E. Oliphant, *A Guide to NumPy*. Trelgol Publishing, 2006, google-Books-ID: fKulSgAACAAJ.

[27] C. Nowke *et al.*, "VisNEST #x2014; Interactive analysis of neural activity data," in *2013 IEEE Symposium on Biological Data Visualization (BioVis)*, Oct. 2013, pp. 65–72.

[28] C. Freitas *et al.*, "On Evaluating Information Visualization Techniques," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '02. New York, NY, USA: ACM, 2002, pp. 373–374.

[29] S. Few, "Data visualization effectiveness profile," 2017. [Online]. Available: https://www.perceptualedge.com/articles/visual_business_intelligence/data_visualization_effectiveness_profile.pdf