Research Article

Engineering: Open Access

ISSN: 2993-8643

Understanding SNN and its Recent Advancements

Fabiha Anan* and Kayes Mohammad Bin Hoque

Department of CSE Brac University

*Corresponding Author
Fabiha Anan, Department of CSE Brac University.

Submitted: 2024, Jun 01; **Accepted**: 2024, Jul 03; **Published**: 2024, Jul 08

Citation: Anan, F., Hoque, K. M. B. (2024). Understanding SNN and its Recent Advancements. *Eng OA, 2*(3), 01-07.

Abstract

The SNNs (spiking neural networks) show what is the nature of computer model of nervous system which is like real brain activity. SNNs might actually be more like brains than the traditional neural networks do because both share an architecture similarity. Lower energy consumption and noise- detecting performance are among their uniqueness. Still, though, current SNNs might seem primitive when compared to the new ones, but their potential to be more powerful and efficient in learning is unlimited. SNNs in many forms are used for numerous functional applications. That entails, for instance, being able to recognize visuals, to perceive language or to take actions. Here, the paper gives a wide view on the SNNs, zooming into their recent successes. The article begins with the saying that neural spiking networks take inspiration of nature exactly from the cause. Another part of the text presents the SNN constituent parts and elicits some SNN advantages compared to the regular neural networks. Subsequently, the text will summarize the latest advancements in SNN, taking into account both hardware and algorithms as well. Another part of the paper will be given to future of sub-national networks.

Keywords: SNN, Spiking Neural Network, Neural Network, DNN

1. Introduction

Machine learning which is being taken to the new height by the application of artificial neural networks is a new advanced application of artificial intelligence. However, in fact a tremendous difference between very sophisticated biological structures that govern human thinking and conceptual levels of AI remains. Let's start with SNNs (spiking neural networks), a category of neural networks designed for this purpose. The purpose of this writing will be not only to explain the basic principles of SNNs but also include latest achievements that put SNNs to the forefront of computational neuroscience. SNNs (spiking neural networks) represent a group of artificial neural networks that emulate the neural functions of the brain with greater fidelity. Unlike conventional Artificial Neural Networks (ANNs) that employ continuous activation func- tions, SNNs (spiking neural networks) utilize discrete spikes to transmit information among neurons. This allows SNNs to monitor changes in biological neural systems over time better, which may further enhance computing power, making it even more robust and powerful. New advancements have been made in different areas such as:

- Creating new neuron models
- Ways to encode information
- Datasets for mimicking the brain
- Algorithms for improving performance
- Frameworks for both software and hardware

Another good example is the NeuCube based system which uses the patterns of the information to handle them more complex than in time and space it is possible. SNN can be used in disciplines like robotics, computer graphics, and natural language processing, alongside the others. As such research will be going on, there will be more astonishing outcomes in the area of SNN.

2. Understanding Spiking Neural Networks (SNNS)

A. What are SNNs?

SNNs are modelled in a similar way to how neurons in our brain transmit signals to each other. Neurons in the brain represent information by sending spikes, which are short electrical signals. The frequency and the exact time when spikes occur is used to encode information. SNNs consists of neurons that are interconnected with each other. Each neuron has a membrane potential, which is the electrical voltage of the cell membrane. Once the membrane potential reaches a critical threshold, the neuron emits a spike. The spike propagates through the network until it reaches subsequent neurons, which may then generate their own spikes.

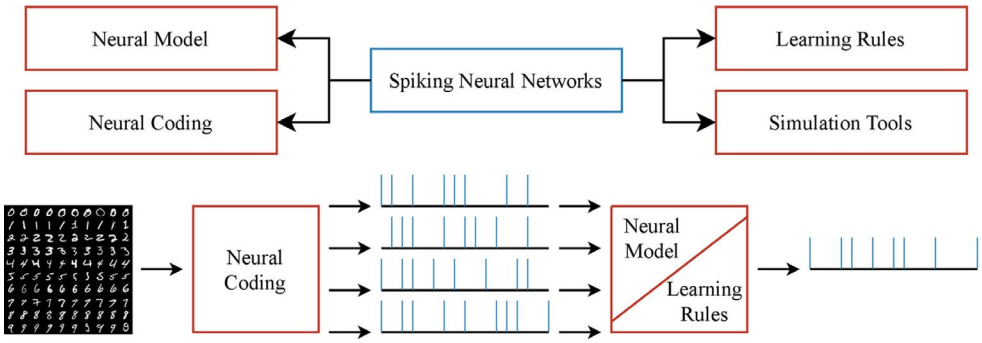

Figure 1: Spiking Neural Networks (SNN) [1].

B. How do SNNs Work?

SNNs operate similarly like the brain. As the Spiking Neural Network (SNN) receives the input, the neurons inside the network start sending electrical signals called spikes. When the intensity and how often they occur are determined by how strong the input is and how the neurons are linked. The sharp signals then move through the connections in the network, and they have the ability to make other neurons send out signals. This iterative process persists until the network reaches a state of stability, characterized by a cessation of significant changes.

The final result of the network is decided by how the neurons in it are firing.

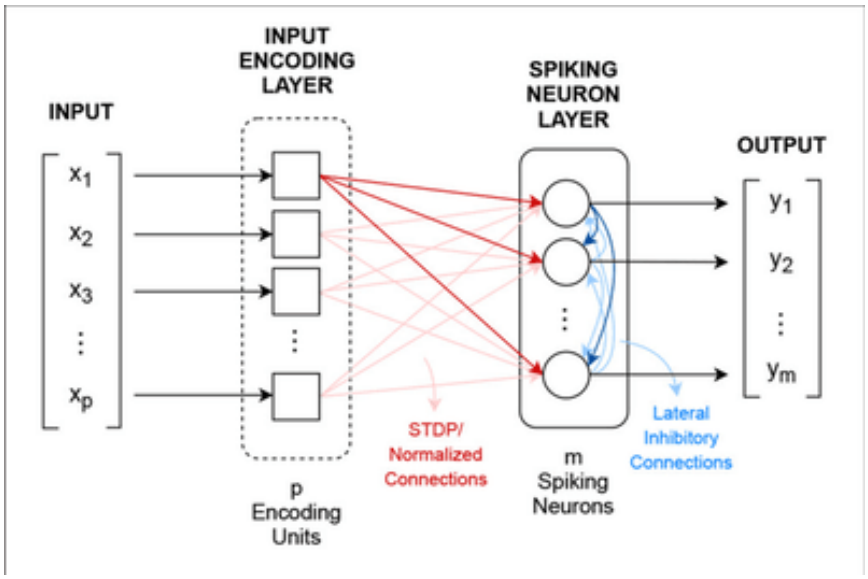

Figure 2: Spiking Neurons [2].

3.Architectures and Frameworks of SNN

Software encompasses a set of instructions and programs that direct a computer on how to execute specific tasks. Software constitutes the intangible component of a computer system that enables the hardware to execute various functions. There are many different ways to create SNNs, but they all have similar goals. Some commonly used frameworks are BindsNET, Norse, Nengo, SpykeTorch, and SNNToolBox. To make SNNs work for general machine learning and reinforce- ment learning tasks, BindsNET was created using PyTorch as its foundation. Norse makes PyTorch better by adding SNNs.

Size	PyTorch	SpiNNker
128	74.7	75.24
256	82.79	82.26
1024	87.96	90.29
2048	90.19	92.36
4096	94.43	93.9

Table I: Current Pytorch and SNN Hidden Layer Comparison

Nengo is constructed using Keras/TensorFlow as its foun- dation and incorporates elements tailored for deep learning, as well as simulation backends such as FPGAs, Loihi, and OpenCL [3]. In simple words, SpykeTorch is a program that helps stimulate convolutional neural networks with one spike per neuron. It also encodes information based on the order of spikes and can learn through a process called STDP or R- STDP. It is based on PyTorch. Lastly, SNNToolBox is a frame- work that helps transform regular artificial neural networks into spiking neural networks (SNNs). It offers different ways to code information and can be used with various simulation tools and hardware platforms such as SpiNNaker and Loihi. It is compatible with popular deep learning libraries like Keras/TF and PyTorch. Many current machine learning frameworks cannot speed up the calculation of binary spike activation and have difficulty handling custom continuous-time differential expressions [4].

4. Gradient Descent Using Sparse Spiking

A method that makes going backwards faster by up to 100 times; they believed that using this method would also save a similar amount of energy. Whenever we are only looking at the neurons that are active, we can completely ignore 98% of the neurons or 98% of the gradient calculation when performing the back-propagation [5]. These accelerations make back prop- agation faster, use less memory, and consume less energy from the GPU without affecting accuracy of the test results. These notable improvements are facilitated by the surrogate gradient, which is utilized to estimate the derivative of the spiking function and becomes more pronounced as the membrane potential nears the threshold. This concentrates most of the gradient on the active neurons.

5. Datasets

Researchers have used the MNIST and CIFAR datasets as trustworthy benchmarks to evaluate and contrast their artificial neural network (ANN) models. Encoding input data as spikes is a necessary step in the implementation of spiking neural networks (SNN). Since event-based sensors capture informa- tion in a manner akin to the brain's information processing, they have gained popularity. Examples of these sensors are ATIS vision sensors. Many people also like to convert regular datasets into neuromorphic form using software.

A. Non-Neuromorphic Datasets

Although neuromorphic datasets are getting more popular, they are not commonly used or following a standard format. This means that many people still have to manually modify normal datasets to make them compatible with spiking inputs. However, there are other well-liked sets of information that people often use, specifically the MNIST and CIFAR-10 databases. The MNIST dataset was used by about 26 people. About 6% of all models, including different versions like N- MNIST and F-MNIST, were used by approximately 33.1%. CIFAR-10 was the second most common dataset, utilized by around 30 people [6]. Interestingly, despite many papers using MNIST, studies using CIFAR tended to have a larger number of models in each study. It's important to note that numer- ous implementations of Spiking Neural Networks (SNNs) in neuroengineering applications are trained and utilized with individual patient data, rather than relying on a standardized benchmark like implanted rhesus monkey EEG data. Due to this, these things were not taken into account during the calculations that were done earlier. We recommend that new researchers use the MNIST dataset or CIFAR-10 to fully benefit from the benchmarking network effect. But, because these two sets of data have already been converted to a type of input that works like our brain, We suggest using the N- MNIST or DVS Gestures datasets that are easy to find and will give consistent results for everyone in the field.

6. Recent Developments

6.1 A. Topology Structures

A key part of spiking neural networks (SNNs) is a layer that is fully connected, recurrent, and convolutional. This layer exhibits resemblances to the layers commonly found in DNNs (deep neural networks). The various types of neu- ral networks include MLPs (multi-layer perceptrons), RNNs (recurrent neural networks), and CNNs (convolutional neural networks). CNNs are frequently employed for tasks involving two-dimensional data, such as images. In contrast, MLPs and RNNs are primarily utilized for handling one-dimensional data, such as text or time series data. By including repeated connections, RNNs can be considered as upgrades to MLPs. This makes them particularly skilled at handling time-related aspects

New Frontiers

Currently, spiking neural networks are not as complicated as the

networks found in living beings. The brain has different levels of connections. To understand how complex network systems work, researchers use multipoint minimum motif networks as the fundamental building blocks of their analysis. Figure 4 demonstrates how different basic patterns can be grouped into 13 categories. This is based on the 3-point pattern and does not consider the different types of nodes, such as different types of neurons [7]. Networks that are similar to each other have consistent and stable patterns in how their motifs are distributed. However, networks that are specifically designed for a certain purpose have noticeable variations in their patterns. We can learn more about how complex biolog- ical networks work by looking at the different patterns and connections found in those networks. Including restrictions in the design of network structures or search methods can be done by using the evaluated motif distributions. This method helps to get new structures that are easier to understand and more likely to exist in biology.

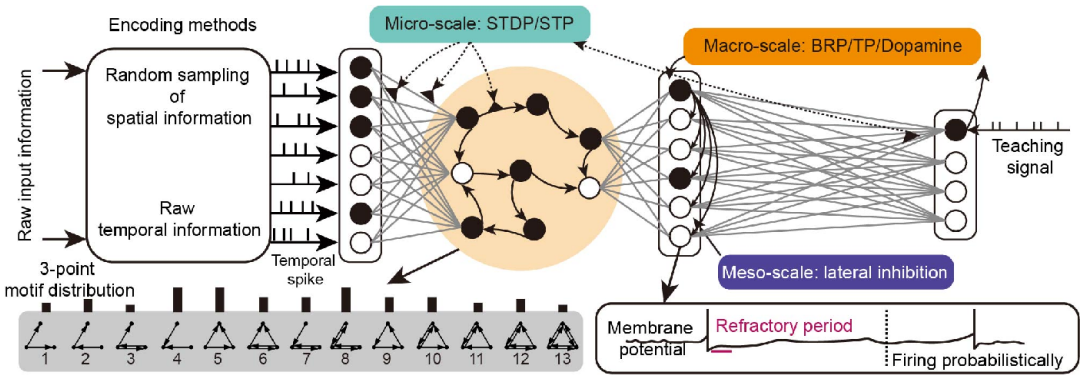

Figure 3: The SNNs Motif Topology, Multi-Scale Synaptic Plasticity, etc [8].

Furthermore, recent methodologies have developed sophis- ticated modeling frameworks that integrate graph neural net- works with spiking neural networks. This fusion allows SNNs to effectively process inputs characterized by graph structures.

6.2 B. Optimization Algorithms

Artificial neural networks (ANNs) are taught using specific sets of information to make the network work as efficiently as possible. In this process, it is important to use optimiza- tion methods. The basis of modern optimization theory for deep neural networks is a method called gradient-based error backpropagation. This method is widely used in real-world applications. Conversely, the field of spiking neural networks (SNNs) lacks a universally agreed-upon fundamental optimiza- tion method. Task performance and biological plausibility have different focuses. The way optimization algorithms work is affected by different ways of representing neurons, encoding information, and the structure of the network. There are two main types of research on improving algorithms for SNNs. The first type doesn't care about how quickly things are done on a computer. Instead, it focuses on using complex models of the brain that are realistic in terms of how the brain works. This helps us learn more about the biological system. The second type focuses on making computers faster by only keeping a few important features and using strange ways to make them work better. The algorithms in the first group attempt to find similarities with what we already know about how living things work. This paper further classifies plasticity optimiza- tion into small scale, medium scale, and large scale categories. Macro-scale plasticity frequently depends on supervised global algorithms, whereas meso and micro-scale plasticity usually make use of self-organizing unsupervised local algorithms. STDP (spike-timing-dependent plasticity) and STP (short- term plasticity) are mechanisms by which individual neurons or synaptic sites can adjust in response to learning stimuli. Reward-STDP, Dale's rule, and other terms are also part of this microscale plasticity.

These algorithms have demonstrated high accuracy in sim- ple sorting with the pictures. For instance, Diehl [2015] made use of STDP to teach and through which they were able to achieve 95% accuracy on the MNIST dataset. By integrating plasticity mechanisms like symmetric STDP and dopamine modifications, the precision was enhanced to 96.7%. [9]. In the article [Kheradpisheh, 2018], author was capable of distinguishing images from the MNIST dataset using multi- layer convolution, STDP, and information delay techniques. They achieved the accuracy of 98.40% [10]. Moreover, the enhancement of multi-layer spiking convolutional networks was proposed through optimization via STDP and Reward- STDP algorithms. Meso-scale plasticity refers to the inter- actions and behaviors exhibited among various synapses and neurons. It consists of lateral inhibition, self-backpropagation, homeostatic circuit control among others. Zhang [2018a] came up with an approach to keep each node's data incoming and outgoing stable [11].

It uses neural homeostasis to optimize the process. Macro- scale plasticity is a method that looks at the big picture when dealing with how credit is distributed worldwide. However, unlike natural networks, there is no agreed-upon universal method for assigning credit that is as effective as backpropaga-tion. The way information is sent between synapses helps fig- ure out if it is

going forward or backward. The weight transport problem is when the brain doesn't have a way to use forward weights while going backwards. Many new algorithms have been created to improve the way backpropagation works. The objective of algorithms like feedback alignment, direct random target propagation, and target propagation is to emulate the functioning of our brains more closely while consuming less energy than traditional backpropagation. These methods use random matrices to transfer gradients directly in the backward process, solving the problem of moving weights. They bring new ideas, like BRP, to help improve the overall efficiency of SNNs. The second set of algorithms often uses different variations of a technique called backpropagation (BP), such as pseudo-BP and SNNs, that have been adjusted to work with deep neural networks (DNNs), in order to optimize a type of neural network called SNN. Using gradient-based BP directly is challenging because it is not possible to differentiate spike signals. By using specific values for the non-smooth parts of spiking neurons, pseudo-BP fixes this issue. Pseudo- BP performs and learns at a similar level to DNNs trained using regular BP on smaller datasets. The idea underlying the transformation from Deep Neural Networks (DNNs) to Spiking Neural Networks (SNNs) revolves around utilizing the average firing frequency of neurons in SNNs to approximate the outputs produced by the Rectified Linear Unit (ReLU) function in DNNs. This is achieved through a distinct method called rate encoding. The first DNNs are trained using BP, and then special methods are used to turn them into SNNs. DNNs that have been converted to SNNs can be used for big networks and datasets. They perform very similar to DNNs with only a small difference in performance. Examples include Rueckauer's enhanced iterations of GoogLeNet and VGG-16 models. According to Sengupta's findings, VGG-16 achieved an accuracy of 69.96% on ImageNet dataset, experiencing a slight loss of 0.56% in conversion precision. Hu subsequently utilized advanced ResNet-50 model, attaining an accuracy rate of 72.75% [12].

New Frontiers

The main goal in improving spiking neural networks (SNNs) is to make them more realistic and efficient. Not many ways can train really big deep SNNs directly, unlike deep neural networks (DNNs). We need to further investigate problems with deep network training, such as when gradients become very small and disappear, the significant amount of resources required for training, and when the network does not converge. Lately, we have been using techniques from the area of DNNs, such as batch normalization and residual learning, to train deep SNNs directly through pseudo-BP. This plan has given us good results and could lead to better optimization in the future for deep SNNs. Moreover, the methods currently employed to convert DNNs into SNNs take a long time to simulate. The range of activation values is drastically reduced during the conversion process, which compresses the model. The concept of DNNs (deep neural networks) is similar to that of BNNs (binary neural networks). However, we do not fully understand the relationship and differences between BNNs and SNNs, as well as how the additional temporal dimension in SNNs can potentially impact their performance. SNNs might be easier to compress because they have a certain way of firing called threshold firing. To make SNNs work even faster, it is important to study how they can be used with compression methods like weight quantization and pruning [2].

6.3 C. Software Frameworks

The SNNs' software framework is a computer program that helps with fast simulation, creating models of networks, and training algorithms. Software frameworks make it easier for people to start working in a specific area and help in building big projects involving SNNs. They provide useful support for the scientific research of SNNs. There are many different software frameworks because research goals and strategies for development are different.

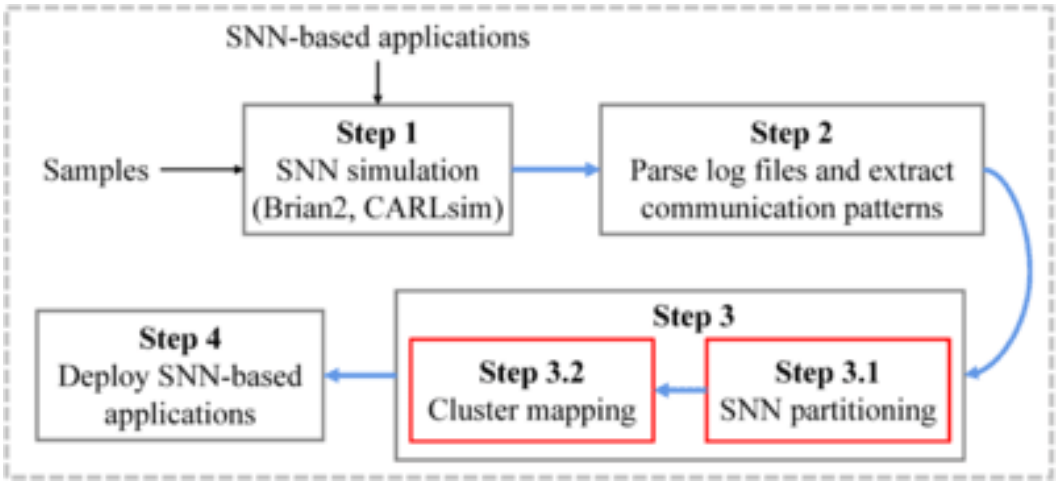

Figure 4: SNN Based Software Working Process [12].

We have been able to create smaller computer models that imitate how the brain works. These models are made using specific computer programs and are mainly used to study liv- ing things. There are two popular software frameworks called Neuron and Nest that are used to support many programming languages, like Python and C++. These frameworks also have visual interfaces [13]. They also support the use of models with complex structures or details about how neurons work, like H-H, LIF, and Izhikevich. Other frameworks can help with different tasks, work with many types of neurons, and support different ways that neurons can change and learn. Bindsnet, Brain2, Spyketorch, SpikingJelly, CogSNN, etc., are software tools created in Python that help with building networks of neurons. They are designed to be good at handling multiple neurons working together and can be used for tasks that involve recognizing complex patterns. In simple terms, CogSNN (a type of computer program) is good at helping with specific sets of data that are related to the brain, like N-MNIST, DVS-CIFAR10, and DvsGesture [14].

New Frontiers

The software frameworks for SNN are still under devel- opment and not completely finished. PyTorch is utilized for constructing and training neural networks, serving as one of the most widely used frameworks for educating about Deep Neural Networks (DNNs). A person new to this can easily create and instruct deep neural networks (DNNs) because of the easy-to-use programming interface and the streamlined data processing method. This greatly enhances the area of DNNs. But, right now, there are only a small number of frameworks that can help with making and training big SNNs. Programmers need to have coding skills to make big SNNs. So, in order to move forward in this field, we have to create programming frameworks that are user-friendly and can effectively handle big SNNs.

6.4 D. Hardware Frameworks

The growth of related applications to more expansive real- world settings is made possible by the development of SNN software frameworks. This benefits areas demanding small size, low energy, and parallel computing, like robotics, pattern recognition, and high-speed cameras. Neuromorphic chips like TrueNorth, Loihi, and Tianjic support SNNs with low energy consumption. These chips have multiple cores working concurrently, each with its own storage, enabling high par- allelism. For facilitating brain-inspired affective computing, IBM's TrueNorth, for instance, features 1 million neurons, 256 million synapses, and 4096 cores. Intel's Loihi enhances olfactory sensitivity with its 8 billion synapses and 8 million neurons. Stanford's Neurogrid provides real-time processing assistance for both high-performance computers and robot cir- cuits that resemble brains [15]. Tsinghua University's Tianjic is a hybrid chip supporting both DNNs and SNNs, demonstrated in applications like speech recognition and obstacle avoidance on self-driving bicycles.

New Frontiers

Neuromorphic chips have become a popular topic of re- search for over 10 years because they offer a different way of computing compared to traditional digital circuits, which can be slow. These chips have had some successful out- comes. Neuromorphic circuits are able to perform tasks with high parallelism and use very little power. They are a good choice for event-driven computation in SNNs because they are designed to work like the brain. Bringing together various useful technologies in computer chips is an important area of research that requires careful examination. These things can work together:

• Combining two different methods, DNNs and SNNs, makes things better.

• Using low-quality memristors and high-quality digital circuits together for calculations.

• Making a computer chip that is good at learning on its own but not so good at big-picture learning.

7. Conclusion

In this paper, We have reviewed the main concepts and latest methods to improve memory and save energy in spiking neural networks (SNNs), while maintaining good performance. This research has reviewed a list of the most common tests being made by the individuals who are learning about SNNs, in addition to their overall performances. In addition, it evaluates new breakthroughs and innovations in SNNs reattuning as well. SNNs will be one of the most significant innovations in the future devices like serious wearables and brain-machine interfaces, which have limited power and storage volume. As we continue to delve into SNNs (spiking neural networks), the third generation of neural networks, we gain deeper understandings of the working of brain and how we can emulate it to create advanced machines. Another plus will be a better outcome of neural networks that will tune at the same time with lower consumption of energy [16,17].

References

1. Shirsavar, S. R., Vahabie, A. H., & Dehaqani, M. R. A. (2023). Models developed for spiking neural networks. *MethodsX, 10,* 102157.
2. Cachi, P. G., Ventura, S., & Cios, K. J. (2021). Crba: A competitive rate-based algorithm based on competitive spiking neural networks. *Frontiers in Computational Neuroscience, 15,* 627567.
3. Rasmussen, D. (2019). NengoDL: Combining deep learning and neuromorphic modelling methods. *Neuroinformatics, 17*(4), 611-628.
4. Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., & Gu, S. (2021). Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems, 34,* 23426-23439.
5. Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., ... & Risbud, S. R. (2021). Advancing neuromorphic computing with loihi: *A survey of results and outlook. Proceedings of the IEEE, 109*(5), 911-934.
6. Rossbroich, J., Gygax, J., & Zenke, F. (2022). Fluctuation-driven initialization for spiking neural network training. *Neuromorphic Computing and Engineering, 2*(4), 044016.
7. Zhu, Z., Peng, J., Li, J., Chen, L., Yu, Q., & Luo, S. (2022). Spiking graph convolutional networks. *arXiv preprint*

arXiv:2205.02767.
8. Zhang, D., Jia, S., & Wang, Q. (2022). Recent advances and new frontiers in spiking neural networks. *arXiv preprint arXiv:2204.07050.*
9. Hao, Y., Huang, X., Dong, M., & Xu, B. (2020). A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule. *Neural Networks, 121,* 387-395.
10. Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., & Masquelier, T. (2019). Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in neuroscience, 13,* 457850.
11. Zhang, D., Zhang, T., Jia, S., Cheng, X., & Xu, B. (2021). Population-coding and dynamic-neurons improved spiking actor network for reinforcement learning. *arXiv preprint arXiv:2106.07854.*
12. Hu, Y., Tang, H., & Pan, G. (2021). Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems, 34*(8), 5200-5205.
13. Migliore, M., Cannia, C., Lytton, W. W., Markram, H., & Hines, M. L. (2006). Parallel network simulations with NEURON. *Journal of computational neuroscience, 21,* 119-129
14. Tielin Zhang and Hongxing Liu. CogSNN. https://github.com/thomasaimondy/CogSNN, 2022. Accessed:2022-05-02.
15. Hao, Y., Huang, X., Dong, M., & Xu, B. (2020). A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule. *Neural Networks, 121,* 387-395.
16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems, 32.*
17. Xiao, C., Chen, J., & Wang, L. (2022). Optimal mapping of spiking neural network to neuromorphic hardware for edge-AI. *Sensors, 22*(19), 7248.