Faculty of Mathematics and Computer Science

Machine learning course (ML)

Recent Advancements in Spiking Neural Networks

Răzvan-Gabriel Petec

Department of Computer Science, Babeş-Bolyai University
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania
E-mail: razvan.petec@stud.ubbcluj.ro

Abstract

Spiking Neural Networks (SNNs), often regarded as the third generation of neural networks [5], provide a biologically inspired approach to computation by leveraging discrete, time-dependent spikes for information processing. These networks offer advantages in energy efficiency and real-time processing, making them suitable for applications in computer vision, robotics, neuromorphic sensing, and healthcare. Recent advancements in training algorithms, such as surrogate gradient techniques and Spike-Timing-Dependent Plasticity (STDP), have improved their performance and scalability. Furthermore, the development of neuromorphic hardware, such as Intel's Loihi [1], has enabled efficient implementations of SNNs for complex tasks like gesture recognition, SLAM, and biomedical signal processing. Despite challenges in training complexity, hardware scalability, and generalization to large-scale datasets, ongoing research continues to address these limitations. This paper explores the architectural foundations, firing functions, learning mechanisms, applications, limitations, and future directions of SNNs, highlighting their potential to revolutionize artificial intelligence with biologically plausible, energy-efficient computation.

Keywords:
Spiking Neural Networks; Neuromorphic Computing; Spike-Timing-Dependent Plasticity; Surrogate Gradient

1. Introduction

The field of artificial intelligence has experienced exponential growth over the past few decades, driven by advancements in deep learning and neural network architectures. While traditional artificial neural networks (ANNs) have demonstrated remarkable success in solving complex tasks, they remain fundamentally distinct from their biological counterparts in terms of structure, energy efficiency, and computational mechanisms. Spiking Neural Networks (SNNs), often referred to as the third generation of neural networks, aim to bridge this gap by mimicking the biological processes of the brain more closely [5].

Unlike conventional ANNs that rely on continuous activations, SNNs utilize discrete spikes to process and transmit information. This temporal dynamic enables SNNs to exploit sparsity and event-driven computation, making them highly energy-efficient compared to traditional deep learning models [7]. These characteristics position SNNs

as a promising alternative for applications requiring low-power consumption and real-time responsiveness, such as autonomous systems, robotics, and edge computing.

As neuromorphic computing – a field inspired by the structure and functionality of the brain – is gaining a lot of interest in the recent years, this has further accelerated the development of SNNs. Neuromorphic hardware, such as Intel's Loihi and IBM's TrueNorth, has introduced novel opportunities for implementing SNNs at scale, enabling efficient learning and inference through spike-based computation [1].

Despite their potential, SNNs face significant challenges, including the complexity of their training processes, the lack of robust learning algorithms, and limited compatibility with current hardware architectures. Nevertheless, ongoing research has made substantial progress in addressing these limitations, with advancements in supervised and unsupervised learning methods, bio-plausible algorithms, and hardware-algorithm co-design [11, 6].

This paper aims to provide a comprehensive overview of recent advancements in SNNs, exploring their architectural foundations, firing functions, and learning mechanisms. It will also be discussed about their applications, limitations, and the future direction of research in this evolving field. By combining insights from biological systems with cutting-edge computational techniques, SNNs have the potential to revolutionize machine learning and expand the boundaries of artificial intelligence.

2. Base architecture

Spiking Neural Networks (SNNs) distinguish themselves from traditional artificial neural networks (ANNs) by using discrete, time-dependent spikes to represent and process information. This dynamic operation not only aligns more closely with biological systems but also allows for efficient, event-driven computation. The underlying architecture of SNNs is built on biologically plausible models of neurons and synapses, and the dynamics are governed by mathematical principles [7].

2.1. Neurons and synapses

2.1.1. Spiking neurons

The behavior of a spiking neuron is commonly modeled by the Leaky Integrate-and-Fire (LIF) framework. The membrane potential $V_m(t)$ evolves over time according to the differential equation:

$$\tau_m \frac{dV_m(t)}{dt} = -V_m(t) + I(t),$$

where:

- τ_m is the membrane time constant,
- $V_m(t)$ is the membrane potential,
- $I(t)$ is the input current.

When $V_m(t)$ exceeds a threshold V_{th}, the neuron fires a spike, and the membrane potential is reset to a baseline value V_{reset}. This behavior can be extended with additional biological features, such as conductance-based synapses or adaptive thresholds, to improve realism [8].

2.1.2. Synapses

Synapses transmit spikes from one neuron to another, with the strength of the connection determined by the synaptic weight w. The postsynaptic current $I_{\text{syn}}(t)$ can be described as:

$$I_{\text{syn}}(t) = \sum_i w_i \cdot \delta(t - t_i),$$

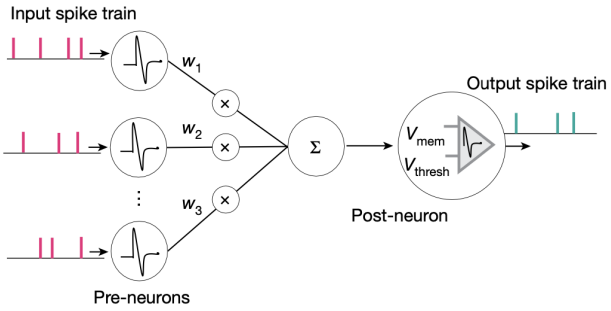

Fig. 1. Spiking Neuron illustration [7].

where $\delta(t - t_i)$ represents a Dirac delta function ($\delta(x) = \{ \infty$ if $x = 0$, else $0 \}$) indicating a spike at time t_i, and w_i is the synaptic weight of the connection. This formulation captures the instantaneous spike transmission and its impact on the postsynaptic neuron [2].

2.2. Network structures

2.2.1. Feedforward SNNs

Feedforward SNNs are composed of layers of spiking neurons where information flows sequentially from input to output. The state of each neuron in the network is influenced by the temporal dynamics of incoming spikes, and the network evolves over time. The membrane potential of neuron j in layer l is:

$$V_m^{(l,j)}(t) = \sum_i w_{ij}^{(l)} \cdot \sigma^{(l-1,i)}(t),$$

where:

- $w_{ij}^{(l)}$ is the synaptic weight connecting neuron i in layer $l - 1$ to neuron j in layer l,
- $\sigma^{(l-1,i)}(t)$ is the spike train from neuron i in the previous layer.

2.2.2. Recurrent SNNs

Recurrent architectures introduce loops within the network, allowing neurons to maintain temporal memory. The dynamics of the recurrent connections are modeled as:

$$V_m^{(t+1)} = f(V_m^{(t)}, \mathbf{W} \cdot \sigma(t)),$$

where $\mathbf{W}$ represents the recurrent weight matrix and $\sigma(t)$ denotes the spike train vector at time t. This architecture is particularly suited for tasks involving temporal patterns [9].

2.3. Temporal coding and representation

In SNNs, information is encoded in the timing and frequency of spikes rather than static activations:

- **Rate coding**: Spike frequency over a time window encodes information, where the firing rate r is:

$$r = \frac{\text{Number of spikes}}{\text{Time window}}.$$

- **Temporal coding**: The precise timing of spikes carries information. For example, latency coding uses the time delay between input and output spikes to represent values. In temporal coding, information is encoded in the precise timing of spikes rather than their frequency or count. The mathematical representation of temporal coding can be expressed as:

$$x_i = f(t_i - t_{\text{ref}})$$

where:
  - x_i is the encoded value or information,
  - t_i is the time of the spike for neuron i,
  - t_{ref} is a reference time (e.g., the stimulus onset time or a baseline spike time),
  - $f(\cdot)$ is a decoding function that maps the spike timing into a meaningful value.
    * In latency coding, earlier spikes represent higher values. For example, a neuron firing closer to the reference time t_{ref} would encode a larger signal. Temporal differences between spikes from multiple neurons can represent more complex data patterns or correlations.
    * Temporal coding schemes exploit the sparse and event-driven nature of spikes to achieve energy efficiency while retaining the ability to process complex data patterns [7].

3. Firing functions

Firing functions in Spiking Neural Networks (SNNs) determine how neurons produce spikes in response to input stimuli. These functions are designed to model the temporal dynamics of biological neurons while maintaining computational efficiency. This section explores key firing models and their mathematical formulations.

3.1. Threshold-Based firing functions

The simplest firing function is threshold-based, where a neuron emits a spike whenever its membrane potential exceeds a predefined threshold. This behavior is expressed as:

$$\sigma(t) = \begin{cases} 1, & \text{if } V_m(t) \geq V_{\text{th}}, \\ 0, & \text{otherwise}, \end{cases}$$

where:

- $\sigma(t)$ is the spike output (1 for a spike, 0 otherwise),
- $V_m(t)$ is the membrane potential at time t,
- V_{th} is the firing threshold.

To incorporate a reset mechanism after firing, the membrane potential is updated as:

$$V_m(t) \leftarrow V_{\text{reset}} \quad \text{if } V_m(t) \geq V_{\text{th}}.$$

This mechanism forms the foundation of many spiking neuron models, including the Leaky Integrate-and-Fire (LIF) model [5].

3.2. Leaky Integrate-and-Fire (LIF) model

The Leaky Integrate-and-Fire (LIF) model introduces dynamics to the membrane potential by combining integration of input currents with a leakage term. The evolution of the membrane potential is governed by:

$$\tau_m \frac{dV_m(t)}{dt} = -V_m(t) + RI(t),$$

where:

- τ_m is the membrane time constant,
- $V_m(t)$ is the membrane potential,
- R is the membrane resistance,
- $I(t)$ is the input current.

The input current is typically modeled as the weighted sum of incoming spikes:

$$I(t) = \sum_i w_i \cdot \delta(t - t_i),$$

where:

- w_i is the synaptic weight of input i,
- $\delta(t - t_i)$ is the Dirac delta function representing a spike at time t_i.

The LIF model is widely used in both theoretical studies and practical applications of SNNs due to its simplicity and biological plausibility [9].

3.3. Adaptive threshold models

To enhance biological realism, adaptive threshold models introduce a time-varying threshold that adjusts based on the neuron's spiking activity. The threshold dynamics are given by:

$$V_{\text{th}}(t) = V_{\text{th},0} + \alpha \sum_k e^{-(t-t_k)/\tau_{\text{th}}},$$

where:

- $V_{\text{th},0}$ is the baseline threshold,
- α is the adaptation constant,
- t_k is the time of the k-th spike,
- τ_{th} is the adaptation time constant.

Adaptive thresholds are crucial for preventing excessive firing and are often implemented in neuromorphic hardware for energy-efficient computation [1].

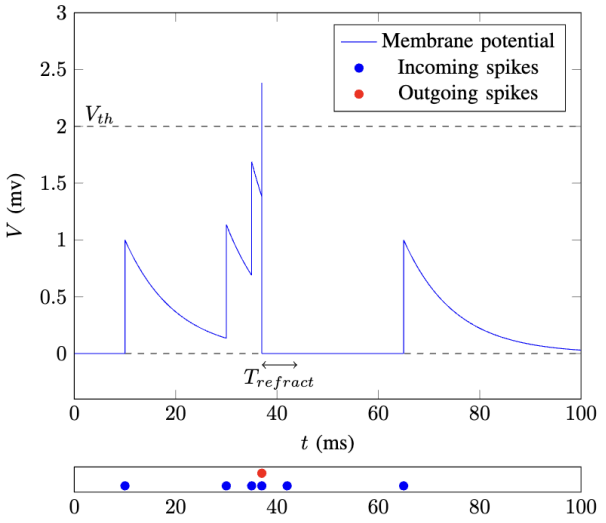

Fig. 2. Leaky Integrate-and-Fire model illustration [3].

4. Learning mechanisms

Learning mechanisms in Spiking Neural Networks (SNNs) determine how synaptic weights are adjusted to achieve desired outputs. These mechanisms can broadly be categorized into **supervised learning**, where the network is guided by a target signal, and **unsupervised learning**, where adjustments occur based on inherent input patterns. Both approaches aim to optimize synaptic weights w_{ij}, which dictate the strength of connections between neurons i and j.

4.1. Supervised learning in SNNs

Supervised learning in SNNs requires a target signal, and the goal is to minimize a loss function $\mathcal{L}$ that quantifies the difference between the predicted output and the desired target. The challenge in SNNs arises from their non-differentiable nature due to the spike generation process, requiring specialized techniques such as surrogate gradients [11].

4.1.1. Loss function

A common loss function used in supervised SNNs is the mean squared error (MSE) for regression tasks:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \left(y_k^{\text{target}} - y_k^{\text{predicted}} \right)^2,$$

where:

- y_k^{target} is the target value for sample k,
- $y_k^{\text{predicted}}$ is the predicted output of the network.

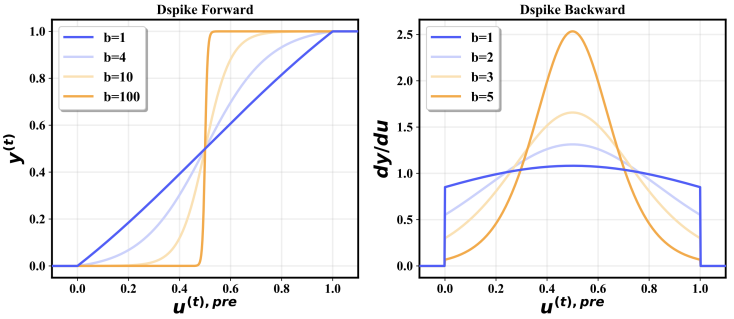

Fig. 3. Surrogate gradient [4]. The forward plot (left) as well as the backward plot (right). By changing temperature parameter we can manipulate the gradient estimation in SNN training.

For classification tasks, a cross-entropy loss is often used:

$$\mathcal{L} = -\sum_{k=1}^{N}\sum_{c=1}^{C} y_k^{\text{target},c} \log\left(y_k^{\text{predicted},c}\right),$$

where C is the number of classes [11].

4.1.2. Weight update rule

Since spike generation is non-differentiable, a surrogate gradient approach is often used to approximate the derivative of the spike function $\sigma(t)$. The surrogate gradient for $\frac{\partial\sigma}{\partial V_m}$ can be defined as:

$$\frac{\partial\sigma}{\partial V_m} \approx \exp\left(-\frac{(V_m - V_{\text{th}})^2}{2\sigma^2}\right),$$

where σ is a smoothing factor [11].

The synaptic weight update is given by:

$$\Delta w_{ij} = -\eta\frac{\partial\mathcal{L}}{\partial w_{ij}},$$

where:

- η is the learning rate,
- $\frac{\partial\mathcal{L}}{\partial w_{ij}}$ is computed using the chain rule through surrogate gradients [11].

4.2. Unsupervised learning in SNNs

Unsupervised learning in SNNs focuses on adjusting synaptic weights based on input correlations or temporal dynamics without external supervision. A widely used mechanism is **Spike-Timing-Dependent Plasticity (STDP)**.

4.2.1. Spike-Timing-Dependent Plasticity (STDP)

STDP adjusts synaptic weights based on the relative timing of spikes between pre- and postsynaptic neurons. The weight update rule is:

$$\Delta w_{ij} = \begin{cases} A_+ e^{-(t_{\text{post}} - t_{\text{pre}})/\tau_+}, & \text{if } t_{\text{post}} > t_{\text{pre}}, \\ -A_- e^{-(t_{\text{pre}} - t_{\text{post}})/\tau_-}, & \text{if } t_{\text{pre}} > t_{\text{post}}, \end{cases}$$

where:

- t_{post} and t_{pre} are the spike times of the post- and presynaptic neurons,
- A_+ and A_- are scaling factors,
- τ_+ and τ_- are time constants for potentiation and depression.

This mechanism has been shown to effectively model biologically plausible synaptic updates [2].

4.2.2. Hebbian learning

A simpler form of unsupervised learning is Hebbian plasticity, where weights are updated based on the co-activation of neurons:

$$\Delta w_{ij} = \eta \, \sigma_{\text{pre}}(t) \cdot \sigma_{\text{post}}(t),$$

where $\sigma_{\text{pre}}(t)$ and $\sigma_{\text{post}}(t)$ are the spike trains of the pre- and postsynaptic neurons [2].

5. Applications

Spiking Neural Networks (SNNs) offer a biologically inspired computational paradigm that has been effectively employed in diverse applications. This section provides an overview of key domains where SNNs have demonstrated their utility, incorporating performance metrics and accuracy figures where available.

5.1. Computer vision

SNNs have been widely used for computer vision tasks, leveraging their ability to process spatiotemporal data from event-driven sensors such as Dynamic Vision Sensors (DVS). Notable applications include:

- **Object recognition**. Using DVS data, SNNs achieve high accuracy with reduced energy consumption. For instance:
  - SNNs trained with Spike-Timing-Dependent Plasticity (STDP) achieved 97.2% accuracy on the MNIST dataset, demonstrating competitive performance with significantly lower power requirements than traditional ANN models [10].
  - On event-based datasets like N-MNIST, SNNs outperform their ANN counterparts in energy efficiency, achieving up to 98.66% accuracy [2].
  - On CIFAR-10-DVS, SNNs trained with surrogate gradient methods achieved accuracy rates exceeding 60%, with improved energy efficiency [11].
- **Gesture recognition**. SNNs, combined with DVS cameras, excel in real-time gesture classification tasks, processing high-temporal-resolution inputs with low energy requirements [10].

5.2. Robotics

In robotics, SNNs provide solutions for control and perception tasks due to their event-driven computation model:

- **Central Pattern Generators (CPGs)**. SNNs modeled as CPGs enable robots to generate biologically inspired locomotion patterns, such as walking and running. These models support adaptive gait generation for hexapod robots and other mobile platforms [10].
- **Navigation and SLAM**. SNNs have been integrated into neuromorphic platforms like Intel's Loihi to efficiently handle tasks such as simultaneous localization and mapping (SLAM). These systems demonstrate real-time performance while consuming significantly less energy compared to conventional robotic control systems [6].

5.3. Neuromorphic sensing

SNNs align naturally with neuromorphic sensors, offering efficient solutions for processing asynchronous and event-driven data:

- **Motion detection**. SNNs, using data from neuromorphic sensors like DVS, excel in motion detection tasks, capturing high-temporal-resolution changes while minimizing power consumption [10].
- **Audio processing**. SNNs have been utilized for auditory scene analysis, speech recognition, and other audio-related tasks, achieving robust performance with lower energy costs compared to traditional methods [7].

5.4. Healthcare

SNNs have shown promise in biomedical applications, particularly for analyzing sequential and noisy temporal data:

- **Brain-Computer Interfaces (BCIs)**. SNNs interpret electroencephalogram (EEG) signals for controlling prosthetic devices, leveraging their ability to detect spatiotemporal patterns. These systems offer real-time decoding capabilities with minimal power requirements [10].
- **Biomedical signal processing**. Applications include seizure detection, arrhythmia classification from electrocardiogram (ECG) data, and electromyography (EMG) analysis. SNN-based approaches achieve comparable accuracy to traditional methods while operating with significantly reduced energy consumption [8].

6. Discussion

6.1. Key metrics and observations

- SNN-based systems demonstrate competitive accuracy on standard datasets, including MNIST (97.2%) and CIFAR-10-DVS (65.61%), while providing energy savings of up to 50× compared to ANN methods [10].
- Neuromorphic hardware implementations, such as Intel's Loihi, support SNN applications with energy consumption as low as 360 nJ per classification on MNIST, further highlighting their efficiency [6].

The versatility and efficiency of SNNs make them a promising choice for domains requiring real-time, low-power processing of spatiotemporal data. Future advancements in neuromorphic hardware and training methodologies are expected to further enhance their applicability.

6.2. Limitations

Despite their promise, Spiking Neural Networks (SNNs) face several challenges that hinder their widespread adoption in practical applications. These limitations span training complexity, hardware constraints, and reduced perfor-

mance on complex tasks compared to artificial neural networks (ANNs). This section elaborates on these issues with insights from the literature.

6.2.1. Training complexity and scalability

Training SNNs is considerably more complex than ANNs due to their non-differentiable nature and reliance on spike-based learning mechanisms like Spike-Timing-Dependent Plasticity (STDP). While surrogate gradient methods alleviate this challenge, they often require careful tuning and substantial computational resources [11].

- **Vanishing spiking activity**. Multi-layer SNNs frequently suffer from a reduction in spiking activity across layers, as highlighted by [3]. This frequency loss undermines training efficiency and often limits SNNs to single-layer architectures for complex tasks.
- **Performance gap with ANNs**: Even with advanced training methods, SNNs generally lag behind ANNs in accuracy for vision datasets like CIFAR-10 and ImageNet, where ANNs achieve over 90% accuracy compared to SNNs' typical range of 60–70% [10].

6.2.2. Hardware constraints

SNNs demand neuromorphic hardware for efficient implementation, as conventional hardware struggles to simulate the event-driven nature of spiking neurons:

- Neuromorphic platforms like Intel's Loihi and SpiNNaker provide significant energy savings, but they are still limited by scalability and memory capacity [10].
- Hardware constraints also restrict the precision and flexibility of SNN implementations. For instance, mirrored STDP proposed by [3] improves learning speed but may face hardware-level challenges in maintaining stable weight distributions.

6.2.3. Latency and coding challenges

Spike-based computation introduces inherent delays due to neural coding schemes:

- **Frequency coding**. While frequency coding provides robust representations, it suffers from high latency and requires large integration windows, making it unsuitable for real-time applications [3].
- **Temporal coding**. Temporal coding reduces latency but is highly sensitive to synaptic delays and parameter settings, particularly in multi-layer architectures [10].

6.2.4. Energy efficiency vs. performance trade-off

While SNNs are often touted for their energy efficiency, this advantage frequently comes at the cost of reduced performance:

- On datasets like MNIST, SNNs can achieve 97% accuracy while consuming significantly less energy than ANNs, but on more complex datasets like CIFAR-10, accuracy often drops below 70%, even with energy-efficient hardware [11].

6.2.5. Limited generalization and dataset constraints

SNNs perform well on simpler datasets like MNIST or DVS-based gesture recognition tasks, but they struggle to generalize to more complex and diverse datasets:

- The reliance on binary or low-resolution input data, such as in frequency or binary coding, limits their applicability to high-dimensional data [3].
- Mixed architectures, such as hybrid SNN-ANN models, have been proposed to address these shortcomings but undermine the energy efficiency of pure SNN designs [7].

7. Conclusions and future work

Spiking Neural Networks (SNNs) represent a transformative approach to computation, bridging the gap between artificial intelligence and the principles of biological neural processing. Their ability to process spatiotemporal data using sparse, event-driven activity makes them uniquely suited for energy-efficient and real-time applications. SNNs hold immense promise in reshaping how we approach tasks in computer vision, robotics, neuromorphic sensing, and healthcare.

One of the most compelling aspects of SNNs is their compatibility with neuromorphic hardware, such as Intel's Loihi [1], which has proven their potential for ultra-low-power computation. These platforms demonstrate that SNNs can achieve substantial energy savings while performing complex tasks like simultaneous localization and mapping (SLAM), gesture recognition, and biomedical signal analysis. Moreover, advances in training methods, such as surrogate gradient techniques and mirrored Spike-Timing-Dependent Plasticity (STDP), are steadily closing the performance gap with traditional artificial neural networks (ANNs).

Despite having promising results, SNNs researchers should address their limitations (Section 6.2), and that requires advancements in both algorithms and hardware:

- **Training improvements**. Developing scalable training methods, such as mirrored STDP [3], and exploring hybrid coding schemes can enhance SNN performance.
- **Hardware development**. Neuromorphic platforms need to scale while maintaining their energy efficiency and supporting multi-layer architectures.
- **Applications focus**. Targeting niche applications like brain-computer interfaces and event-driven robotics, where SNNs' energy efficiency and temporal processing excel, may accelerate their adoption.

While SNNs hold promise for energy-efficient, biologically plausible computation, overcoming these limitations is essential for achieving their full potential.

Declaration of Generative AI and AI assisted technologies in the writing process

During the preparation of this work the author used ChatGPT 4o in order to improve the language and readability of their paper. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

References

[1] Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al., 2018. Loihi: A neuromorphic manycore processor with on-chip learning. Ieee Micro 38, 82–99.

[2] Diehl, P.U., Cook, M., 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience 9, 99.

[3] Falez, P., Tirilly, P., Bilasco, I.M., Devienne, P., Boulet, P., 2018. Mastering the output frequency in spiking neural networks, in: 2018 international joint conference on neural networks (IJCNN), IEEE. pp. 1–8.

[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] Maass, W., 1997. Networks of spiking neurons: the third generation of neural network models. Neural networks 10, 1659–1671.

[6] Rathi, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., Roy, K., 2023. Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. ACM Computing Surveys 55, 1–49.

[7] Roy, K., Jaiswal, A., Panda, P., 2019. Towards spike-based machine intelligence with neuromorphic computing. Nature 575, 607–617.

[8] Sanaullah, Koravuna, S., Rückert, U., Jungeblut, T., 2023. Exploring spiking neural networks: a comprehensive analysis of mathematical models and applications. Frontiers in Computational Neuroscience 17, 1215824.

[9] Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., Masquelier, T., Maida, A., 2019. Deep learning in spiking neural networks. Neural networks 111, 47–63.

[10] Yamazaki, K., Vo-Ho, V.K., Bulsara, D., Le, N., 2022. Spiking neural networks and their applications: A review. Brain Sciences 12, 863.

[11] Zenke, F., Ganguli, S., 2018. Superspike: Supervised learning in multilayer spiking neural networks. Neural computation 30, 1514–1541.