A Survey of Spiking Neural Networks and Their Learning Strategies

Md Asiful Hoque Prodhan, The University of Texas at San Antonio

Abstract

Artificial Neural Networks (ANNs) are well-known for their capacity to effectively process massive amounts of data and solve complex problems through parallel computation. They can learn from experience and continue to improve their accuracy and performance as they are exposed to new data over time. Despite these strengths, ANNs often require significant computational power and energy, especially when deployed on edge devices or in real-time systems. They also lack the ability to naturally process time-dependent information the way biological brains do. Spiking Neural Networks (SNNs) overcome these limitations by using a communication method that resembles how the brain works. Instead of constant activation, they send signals only at specific times, which makes them more energy-efficient and better suited for processing time-based information. This survey presents a structured overview of key research in SNNs, covering their learning and training methods. The paper also discusses current challenges, including training complexity, scalability, and accuracy gaps, and identifies future research directions for the field.

Keywords: spiking neural networks, artificial neural networks, neural network, deep learning, learning strategies, biological mechanism, survey, review.

Table of Contents

1. Introduction

Spiking Neural Networks mark a major advancement in the domain of neural networks, drawing inspiration from the biological systems of the brain. In contrast to conventional Artificial Neural Networks that use continuous activation functions, SNNs employ discrete spikes to communicate among neurons. Therefore, SNNs provide a computational model that is more biologically plausible. This unique approach allows SNNs to efficiently handle temporal data, making them suitable for use in applications that require real-time computation with minimal power consumption. As edge computing and neuromorphic hardware become more prevalent, the energy-saving and real-time computation abilities of SNNs make them a promising solution for various AI tasks, such as computer vision, speech recognition and robotics.

Despite the potential advantages, the development of SNNs has faced significant challenges. Training these networks is particularly difficult because of the non-differentiable nature of spiking events, which hinders the application of gradient-based learning methods typically employed in traditional deep learning models. Furthermore, SNNs have historically struggled to scale effectively on conventional hardware platforms, limiting their adoption in large-scale applications. These challenges have led to extensive research in recent years, focusing on improving the efficiency of SNNs through innovative training algorithms, hardware acceleration, and hybrid models that leverage the strengths of both traditional ANNs and SNNs.

The purpose of this survey is to provide an organized review of the primary research on SNNs, with an emphasis on their learning strategies and training methods. The paper is organized as follows: Section 2 introduces the foundations of SNNs and a few basic terminologies discussed throughout this survey. Section 3 explores various neuron models used in SNNs and input encoding methods. Section 4 covers the different learning and training techniques (Diehl & Cook, 2015; Jin et al., 2018; Lee et al., 2016; Neftci et al., 2019; Rathi et al., 2020; Rueckauer et al., 2017; Sengupta et al., 2019; H. Wu et al., 2021; Y. Wu et al., 2018) in SNNs. In Section 5, the paper discusses the current challenges faced by SNNs, followed by future research directions intended to address these issues. Finally, Section 6 concludes the paper with a summary of the findings and highlights the significance of the survey.

2. Foundations of Spiking Neural Networks

2.1. Evolution of Neural Networks

SNNs are recognized as the third generation of neural networks, a classification introduced by Maass (1997). The first generation, represented by McCulloch-Pitts neurons, used digital outputs to perform logical operations, forming the basis of neural computation. The second generation utilizes continuous activation functions (for example, the sigmoid and linear saturated functions), which are used to weighted sums of inputs. These methods allow the creation of more complex networks, including feedforward and recurrent networks, capable of computing both boolean and analog functions. However, these models still lack the ability to replicate the temporal behavior of biological neurons. The third generation, represented by spiking neurons, models brain-like behavior by information encoded in the exact timing of spikes.

2.2. Emergence of SNNs as the Third Generation of Neural Networks

In contrast to the earlier generations of neural networks, SNNs utilize the precise timing of each spike for encoding information, a feature that provides a more biologically realistic representation of neural computation (Ghosh-Dastidar & Adeli, 2009). The development of these models emerged from the limitations of first- and second-generation networks, which neglected the precise temporal information carried by spikes. While the second generation used activation functions to approximate firing rates, SNNs capture the temporal dynamics of neurons more accurately, mimicking the action potentials or spikes in biological neurons, which are critical in brain function. This temporal encoding through spike trains significantly enhances the computational capacity of SNNs, which allows them to handle advanced time-dependent activities, including real-time pattern recognition and processing sensory information.

2.3. Terminologies & Basic Concepts

Before diving into SNNs, the key terms and concepts that will be discussed throughout this survey are first defined.

Term	**Definition**
Neuron	A unit that integrates incoming signals and, when sufficiently activated, produces a spike.
Spike (Action Potential)	A brief electrical pulse emitted by a neuron to signal its activation to other neurons. It is a binary event; either a full spike occurs or nothing.
Membrane potential	The neuron’s internal voltage that rises and falls in response to inputs.
Threshold	The voltage level that must be reached by the membrane potential for the neuron to generate a spike.
Firing	The process of a neuron emitting a spike once its threshold is reached.
Resting potential	The baseline voltage of a neuron when it is not receiving any input.
Refractory period	A short duration after firing, during which the neuron cannot fire again
Synapse	The connection point where one neuron’s spike influences another neuron’s membrane potential.
Presynaptic potential	The change in voltage at the sending neuron’s synapse that triggers the release of its spike signal.
Postsynaptic potential	The change in a neuron’s membrane potential caused by a spike from another neuron through the synapse.
Leak	The gradual decay of a neuron’s voltage back to its resting potential when it is not receiving input.
Time-step	A small moment in time when all neurons check inputs and update their voltages.
Event-driven	A mode where neurons update only when spikes occur, rather than at every timestep.

Asynchronous	A strategy where neurons operate independently, without requiring a global clock to synchronize updates.
Event queue	A list of scheduled spike events waiting to be processed by the network.

3. Neuron Models and Signal Encoding in SNNs

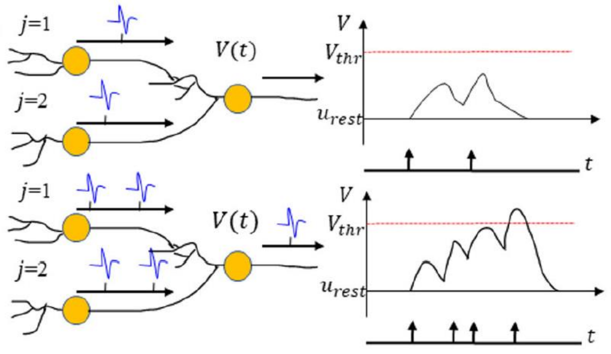
3.1. Neuron Models

Neurons can be modeled as simple electrical circuits made of a resistance (R) and a capacitor (C) (Niu et al., 2023). The capacitor represents the cell membrane storing charge, and the resistance indicates the membrane's leakage. By Kirchhoff's Current Law, when an external input current (t) arrives, it splits into I_R and I_C, so $I(t) = I_R + I_C$. Here, I_R is the passing flowing through the leak resistance R and I_C is the current charging the membrane capacitor C.

As neurons get incoming spikes, the membrane potential $V(t)$ rises, As illustrated in Figure 1. When $V(t)$ exceeds the firing threshold V_{thr}, the neuron releases an action potential (spike) and its membrane potential is reset toward the resting potential u_{rest} (J. Wu et al., 2019).

Figure 1

The integration of membrane potential and the firing of spikes in spiking neurons



Note. From "Research Progress of Spiking Neural Network in Image Classification: A Review" by L. Y. Niu, Y. Wei, W. B. Liu, J. Y. Long, & T. H. Xue, 2023, *Applied Intelligence*, *53*(16), 19466–19490 (https://doi.org/10.1007/s10489-023-04553-0). Copyright 2023 by the author(s).

3.1.1. Integrate-and-fire (IF) Model

In the IF model, the neuron's membrane simply integrates incoming current without any leak (Niu et al., 2023). Its voltage $V(t)$ follows the equation:

$$\frac{dV(t)}{dt} = \frac{1}{C} I(t) \tag{1}$$

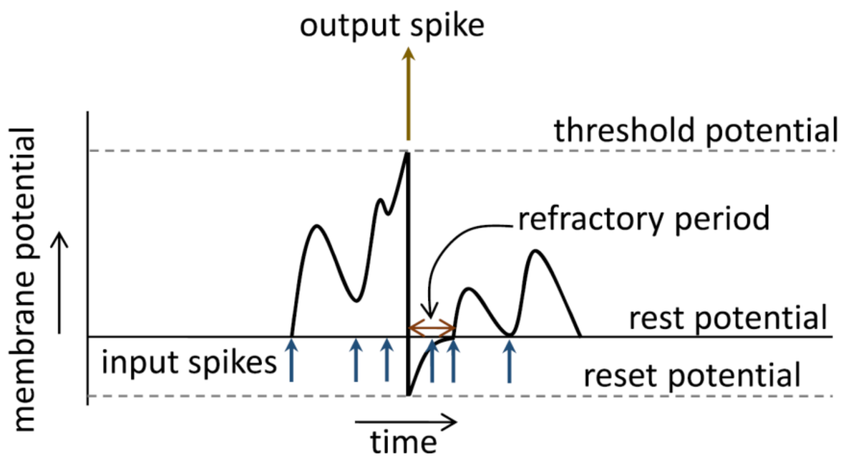
The membrane potential of the neuron increases steadily in response to input current until it exceeds the threshold for firing and triggers a spike. Without any input current, the voltage remains fixed at its last value (Cruz-Albrecht et al., 2012).

3.1.2. Leaky Integrate-and-Fire (LIF) Model

In the LIF model, the membrane potential not only integrates current input but also decays toward its resting value when no input is present (L. Zhang et al., 2019), as shown in Figure 2. After a spike, there is a refractory period when the membrane potential does not accumulate the input current, preventing continuous firing of the neuron (Rathi et al., 2023).

Figure 2

Effects of a LIF model in reaction to input spikes



Note. From “Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware” by N. Rathi, I. Chakraborty, A. Kosta, A. Sengupta, A. Ankit, P. Panda, & K. Roy, 2023, *ACM Computing Surveys, 55*(12), 1–49 (https://doi.org/10.1145/3571155). Copyright 2023 by the author(s).

The model follows:

$$\tau_m \frac{dV}{dt} = -(V(t) - u_{\text{rest}}) + R\,I(t) \qquad (2)$$

where $\tau_m = RC$ is the membrane time constant and u_{rest} is the resting potential (Niu et al., 2023). If $V(t)$ stays below the firing threshold, the neuron does not spike and $V(t)$ decays toward u_{rest} at a rate set by τ_m. Compared to the IF model, the added “leak” term captures the gradual loss of membrane charge seen in real neurons.

3.1.3. Hodgkin-Huxley (HH) Model

The HH model, based on experiments with the squid giant axon, revealed that action potential generation depends mainly on sodium (Na^+) and potassium (K^+) ion channels (Hodgkin & Huxley, 1952). The membrane potential $v_m(t)$ changes over time according to

$$C_m \frac{dv_m(t)}{dt} = I_{\text{ion}}(t) + I_{\text{syn}}(t) \tag{3}$$

where C_m is the membrane capacitance, $I_{\text{syn}}(t)$ is the synaptic input current, and $I_{\text{ion}}(t)$ represents the ionic currents flowing through K^+, Na^+, and leak channels (Yamazaki et al., 2022).

The ionic current is expressed as

$$I_{\text{ion}}(t) = G_K n^4 (v_m - E_K) + G_{Na} m^3 h (v_m - E_{Na}) + G_L (v_m - E_L) \tag{4}$$

where G_K, G_{Na}, and G_L denote the maximum conductance of the K^+, Na^+, and leak channels, respectively, and E_K, E_{Na}, and E_L are their associated reversal potentials. These channels open and close according to three voltage-dependent variables, *n*, *m*, and *h*, that adjust based on the membrane potential. These variables directly influence the ionic currents, where *n* affects potassium channel activation, and *m* and *h* regulate sodium channel behavior.

The HH model naturally reproduces how the membrane potential changes during an action potential without requiring artificial reset mechanisms used in simpler models like LIF. However, due to its complexity, the HH model requires substantial computational resources and is not practical for simulating extensive spiking neural networks.

3.2. Input Encoding

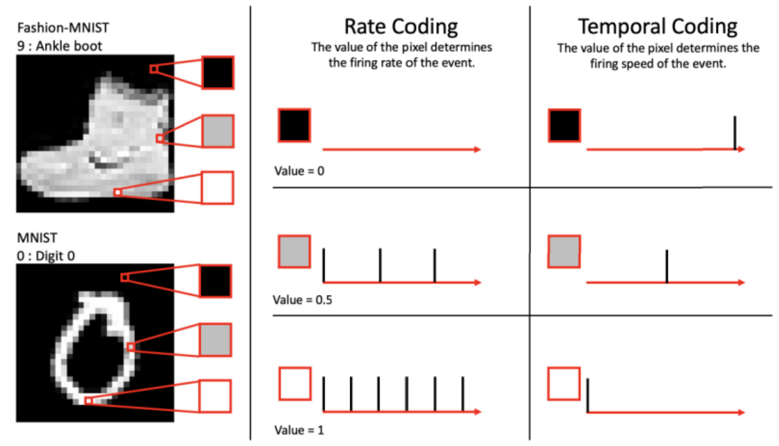
In SNNs, continuous inputs are encoded into discrete spikes before computation. Common encoding schemes, as shown in Figure 3, include rate encoding and several forms of temporal encoding.

3.2.1. Rate Encoding

Rate encoding conveys information through the average firing rate of a neuron over a specified time window, with the exact timing of each spike being irrelevant (Rathi et al., 2023). A common approach, known as Poisson encoding (Diehl & Cook, 2015; Rueckauer et al., 2017), generates spikes by comparing the normalized analog value against a random number at each timestep; the neuron emits a spike (1) when the analog value exceeds the random number, otherwise it remains silent (0) (Rathi et al., 2023). Rate encoding tends to be suboptimal, as each spike carries only a small amount of information.

Figure 3

Rate Encoding and Temporal Encoding Methods


Fashion-MNIST
9 : Ankle boot
MNIST
0 : Digit 0
Rate Coding
The value of the pixel determines the firing rate of the event.
Temporal Coding
The value of the pixel determines the firing speed of the event.
Value = 0
Value = 0.5
Value = 1

Note. Rate Encoding (left): In this method, pixel values are converted into spike rates. Higher pixel values lead to higher firing rates, with the quantity of spikes within a given time window reflecting the intensity of the pixel. Temporal Encoding (Right): In this approach, pixel values are represented by the precise timing of spikes. Higher pixel values result in earlier spikes, while lower pixel values cause spikes to occur later. From “Event-based Spiking Neural Networks for Object Detection: Datasets, Architectures, Learning Rules, and Implementation” by C. Iaboni & P. Abichandani, 2024, *IEEE Access*, *12*, 180532–180596 (https://doi.org/10.1109/ACCESS.2024.3479968). Copyright 2024 by the author(s).

3.2.2. Temporal Encoding

Temporal encoding conveys information based on the exact timing of spikes, where more prominent or important features are represented by earlier spikes (Yamazaki et al., 2022). Unlike rate encoding, which relies on spike frequency, temporal encoding uses spike timing, leading to much sparser activity. This method allows input features to be represented by small neuron groups, but it is more sensitive to noise and timing variability. Several widely adopted encoding methods rely on temporal encoding (Tan et al., 2020).

- **Time-to-First-Spike:** Here, information is represented through the delay from the start of the stimulus to the first spike, allowing rapid decision-making within milliseconds.
- **Rank Order Coding:** Here, information is carried by the order in which a group of neurons fires, with each neuron spiking only once after stimulus presentation (Thorpe & Gautrais, 1998).
- **Latency Coding:** Latency coding encodes information through the relative timing between spikes, where timing differences below 20 ms lead to long-term potentiation (LTP), and larger differences cause long-term depression (LTD).

- **Phase Coding:** In phase coding, neurons encode information by the phase of their spikes with respect to a reference oscillation, a phenomenon frequently seen in various regions of the brain (Buzsáki, 2006).
- **Synchrony Coding:** Synchrony coding represents information by neurons firing simultaneously when they correspond to different features of the same object (Buzsáki, 2006).

4. Learning and Training Methods in SNNs

Learning in SNNs is typically divided into supervised, unsupervised, and indirect learning strategies (Iaboni & Abichandani, 2024). Supervised learning in SNNs often adapts traditional techniques such as backpropagation and surrogate gradient methods to accommodate the non-differentiability of spike events (Bohte et al., 2000; Lee et al., 2016; McKennoch et al., 2006; Wang et al., 2020; Xu et al., 2013). It adapts conventional gradient-based optimization methods to allow the network to learn from labeled data. On the other hand, unsupervised learning typically utilizes Spike-Time-Dependent Plasticity (STDP) (Legenstein et al., 2005), where synaptic weights are adjusted depending on the spike timing between presynaptic and postsynaptic neurons (Diehl & Cook, 2015; Masquelier & Thorpe, 2007). In indirect learning, an Artificial Neural Network is first trained using conventional methods before converting it into an SNN (Iaboni & Abichandani, 2024). This approach takes advantage of the computational efficiency of SNNs while retaining the benefits of the ANN's pre-training (Hunsberger & Eliasmith, 2015; Rathi et al., 2020; Rueckauer et al., 2017; Rueckauer & Liu, 2018; Sengupta et al., 2019).

4.1. Spike-Time-Dependent Plasticity

STDP (Legenstein et al., 2005) is a biologically feasible, unsupervised learning approach (Markram et al., 2012) founded on Hebbian principles (Ruf & Schmitt, 1997). According to this rule, when two connected neurons fire simultaneously, the synaptic weight between them should be strengthened (Song et al., 2000). In STDP, the adjustment of synaptic weights depends on the precise order of spikes: if a presynaptic spike happens prior to a postsynaptic spike, the synapse may either be strengthened or weakened (Pietrzak et al., 2023). Conversely, if a postsynaptic spike happens earlier than a presynaptic spike, the synapse may also be either strengthened or weakened, depending on the specific learning rule. When the first case leads to strengthening and the second to weakening, it is called Hebbian STDP; If the weight changes in the reverse manner, it is known as anti-Hebbian STDP.

The rule that determines how much a synaptic weight is adjusted in STDP is referred to as the learning window (Pietrzak et al., 2023). A common Hebbian learning (Ruf & Schmitt, 1997) window is defined by Equation (5), where if the presynaptic spike t_{pre} occurs before or at the same time as the postsynaptic spike t_{post}, the synaptic weight is strengthened by a factor proportional to $A_{+}\exp\left(\frac{t_{\text{pre}}-t_{\text{post}}}{\tau_{+}}\right)$ (Pietrzak et al., 2023). If the presynaptic spike happens after the postsynaptic spike, the synapse gets weakened by $A_{-}\exp\left(-\frac{t_{\text{pre}}-t_{\text{post}}}{\tau_{-}}\right)$. In contrast, an anti-

Hebbian window, described in Equation (6), reverses this behavior: if $t_{\text{pre}} \geq t_{\text{post}}$, the synapse is potentiated, while if $t_{\text{pre}} < t_{\text{post}}$, it is depressed.

$$\Delta = \begin{cases} A_{+}\exp\left(\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{+}}\right), & \text{if } t_{\text{pre}} \leq t_{\text{post}} \\ A_{-}\exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{-}}\right), & \text{if } t_{\text{pre}} > t_{\text{post}} \end{cases} \tag{5}$$

$$\Delta = \begin{cases} A_{+}\exp\left(\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{+}}\right), & \text{if } t_{\text{pre}} \geq t_{\text{post}} \\ A_{-}\exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{-}}\right), & \text{if } t_{\text{pre}} < t_{\text{post}} \end{cases} \tag{6}$$

In these equations, A_{+} and A_{-} are scaling factors, respectively controlling the degree of synaptic potentiation and depression. t_{pre} and t_{post} denote the spike times for presynaptic and postsynaptic, while τ_{+} and τ_{-} are the time constants associated with potentiation and depression.

Diehl and Cook (2015) presented a biologically plausible SNN that achieved unsupervised digit recognition using STDP. Their network consists of conductance-based synapses, lateral inhibition, and adaptive thresholds to regulate neuron firing rates. Inputs were encoded as Poisson-distributed spike trains derived from pixel intensities. Without the use of labeled data, the model obtained 95% accuracy on the MNIST dataset with the BRIAN simulator.

The study demonstrated that their network was robust across four different STDP rules, suggesting that the combination of mechanisms allows the network to learn prototypical inputs effectively (Diehl & Cook, 2015). The network showed consistent performance scaling as the number of neurons increased. It offered consistent results across different STDP learning rules, suggesting its potential for broader applications in neuroscience and machine learning.

4.2. Spike-Based Backpropagation

The paper by Lee et al. (2016) introduced a novel technique to training deep SNNs using backpropagation, a method traditionally employed in Artificial Neural Networks (ANNs). The key challenge addressed by the authors is the non-differentiable characteristic of spike events in SNNs. To overcome this, they reformulated the membrane potentials by utilizing low-pass filtered spike signals, treating spike-time discontinuities as noise to maintain differentiability. This modification allowed the application of gradient-based optimization techniques like SGD or ADAM (Kingma & Ba, 2014) directly on spikes.

The study proposed a framework that incorporates fully connected and convolutional SNNs, where the backpropagation equations are modified to suit LIF neurons (Lee et al., 2016). This framework includes additional techniques like weight initialization, threshold regularization, and lateral inhibition through winner-takes-all (WTA) circuits to stabilize training and improve learning efficiency. The method had been tested on the MNIST and N-MNIST datasets and it obtained high accuracy (98.77% on MNIST and 98.66% on N-MNIST), on par with traditional deep neural networks (DNNs). The main strength of this method was its capability to train deep

SNNs from scratch using true spike-based inputs, significantly reducing computational costs compared to ANNs.

The paper by H. Wu et al. (2021) introduced a novel backpropagation method called Accumulated Spiking Flow Backpropagation (ASF-BP) to train SNNs. The method replaced spike-train-based gradients with accumulated spiking flows and built an equivalent network. It then applied a single-loop backpropagation based on these accumulated values, using adaptive scale factors to match actual neuron behavior. ASF-BP reduced computational complexity by aggregating the inputs and outputs of spiking neurons across time steps instead of relying on spike trains. It avoided the vanishing gradient problem and simplified backward computations.

Y. Wu et al. (2018) proposed an STBP (Spatio-Temporal Backpropagation) algorithm for training SNNs, which includes both temporal and spatial aspects to improve performance. The authors developed a recursive form of the LIF model to support gradient-based optimization in both spatial and temporal domains. They introduced an STBP algorithm that performs error backpropagation by unfolding the spiking neuron states across the spatial (layer-based) and temporal (timestep-dependent) dimensions. To address spike non-differentiability, they proposed four approximations (rectangular, polynomial, sigmoid, and Gaussian curves) to approximate the gradients. It outperformed many state-of-the-art models while avoiding complex techniques such as error or weight normalization during training.

Jin et al. (2018) introduced an HM2-BP (hybrid macro/micro-level backpropagation) method for deep SNN training. S-PSPs (Spike-train-level post-synaptic potentials) are used to capture temporal dynamics at the micro-level, while rate-coded errors are backpropagated at the macro-level to guide learning. By directly calculating the rate-coded loss function’s gradient, this technique enabled effective training of deep SNNs.

4.3. Surrogate Gradient Learning in SNNs

The paper by Neftci et al. (2019) discussed how difficult it is to train SNNs due to their binary and dynamic nature, particularly in deep architectures with hidden layers. To address these issues, the authors introduced Surrogate Gradient (SG) methods, which enable the non-smooth spiking nonlinearity to be continuously relaxed. It permitted the application of gradient-based optimization in SNNs. These methods were applied in various contexts, such as feedback alignment, local three-factor learning rules like SuperSpike (Zenke & Ganguli, 2018), and spike-time-based learning, offering efficient and scalable training solutions for SNNs. The authors first reformulated SNNs as a special case of RNNs and then applied surrogate gradient methods to allow gradient-based learning. They used approximations of the spiking nonlinearity with smooth functions (e.g., fast sigmoid, exponential) to enable backpropagation.

SG methods allowed the training of deep SNNs with hidden layers, solving spatial and temporal credit assignment problems (Neftci et al., 2019). These strategies can be adopted in modern ML frameworks and support both online and offline learning.

4.4. Conversion of Artificial Neural Networks to Spiking Neural Networks

Sengupta et al. (2019) introduced a conversion technique from deep Analog Neural Networks (ANNs) to SNNs using a novel weight-normalization approach called Spike-Norm. This technique ensured minimal accuracy loss during conversion by considering the actual spiking behavior of SNNs, unlike previous methods that relied solely on ANN activations. The authors showed that their method works on challenging datasets like CIFAR-10 and ImageNet through deep architectures like VGG and ResNet, getting state-of-the-art SNN performance. The study used a data-driven weight normalization strategy during ANN-to-SNN conversion, where neuron thresholds are adjusted according to the behavior of actual SNN spikes. They applied this Spike-Norm method layer by layer, starting from Poisson-encoded inputs and propagating spike trains through VGG-16 and ResNet networks. The model operated without bias terms or batch normalization and uses dropout for regularization.

Rueckauer et al. (2017) presented a conversion strategy from traditional analog deep networks to SNNs to classify images. The conversion process allowed nearly arbitrary CNN architectures to be transformed into SNNs. The authors developed a theoretical basis for converting ANN activations to equivalent SNN spike rates. They replaced ANN components with spike-based counterparts such as softmax, batch normalization, and max-pooling. They introduced a robust normalization scheme based on percentiles to improve firing rate estimates and reduce error. The conversion was applied to full-scale architectures like VGG-16 (Simonyan & Zisserman, 2014) and Inception-v3 (Szegedy et al., 2016) using their custom SNN toolbox. The empirical results revealed that SNNs have significantly lower computing costs than traditional ANNs. For example, the SNN version of LeNet (Lecun et al., 1998) on MNIST achieved a 2x reduction in operations with only a slight increase in error rate.

Rueckauer and Liu (2018) proposed a technique to convert Analog Neural Networks (ANNs) to SNNs utilizing a temporal coding scheme focused on the time-to-first-spike (TTFS). The TTFS method represented ANN neuron activations as inverse spike times in SNNs, reducing the number of spikes and computational costs significantly. The authors trained a LeNet-5 (Lecun et al., 1998) ANN on MNIST and converted it to an SNN using TTFS encoding. Three variants of the TTFS method were proposed: a baseline version, one with dynamic thresholds, and another using a clamped ReLU activation function. In the base model, neurons spike once, and the timing represents the inverse of their activation value. The dynamic threshold variant adjusted firing thresholds based on incoming spikes, while the clamped version retrained the ANN using a modified ReLU function to minimize the impact of long-latency spikes. The TTFS method achieved a 7-10X reduction in operations on the MNIST dataset with less than a 1% drop in accuracy in comparison to the original ANN.

The work by Rathi et al. (2020) presented a hybrid approach to training SNNs by integrating artificial-to-spiking network conversion with STDB (Legenstein et al., 2005). The authors began by training an ANN, which was later converted to an SNN using threshold balancing (Rathi et al., 2020). Fine-tuning of the resulting SNN was then performed using STDB, a surrogate gradient method that uses the time since a neuron’s last spike. The forward pass accumulated membrane potentials, and gradients were computed at each time step using spike timing. Dropout was used instead of batch normalization, and average pooling was preferred to preserve information across layers. This method minimized the inference time steps by 10-25x compared to purely converted SNNs, achieving similar accuracy while significantly lowering training complexity.

4.5. Analysis of SNNs Performance

Table 1 demonstrates how different SNN models performed on image classification tasks across a variety of image datasets (Rathi et al., 2023). The best results are obtained with backpropagation techniques.

Table 1

SNNs Performance Evaluation on Image Recognition Datasets

Datasets	Paper	Neuron model	Input coding	Learning rule	Accuracy
MNIST	(Diehl & Cook, 2015)	LIF	Rate	STDP	95%
	(Rueckauer & Liu, 2018)	IF	Temporal	ANN-to-SNN	98.53%
	(W. Zhang & Li, 2020)	LIF	Encoding layer	Backpropagation	99.53%
CIFAR-10	(Sengupta et al., 2019)	IF	Rate	ANN-to-SNN	91.55%
	(Rueckauer et al., 2017)	IF	Rate	ANN-to-SNN	90.85%
	(Rathi et al., 2020)	LIF	Rate	Hybrid	92.02%
	(W. Zhang & Li, 2020)	LIF	Encoding layer	Backpropagation	91.41%
ImageNet	(Sengupta et al., 2019)	IF	Rate	ANN-to-SNN	69.96%
	(Rueckauer et al., 2017)	IF	Rate	ANN-to-SNN	49.61%
	(Rathi et al., 2020)	LIF	Rate	Hybrid	65.19%

Note. Adapted From “Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware” by N. Rathi, I. Chakraborty, A. Kosta, A. Sengupta, A. Ankit, P.

Panda, & K. Roy, 2023, *ACM Computing Surveys*, *55*(12), 1–49 (https://doi.org/10.1145/3571155). Copyright 2023 by the author(s).

5. Challenges and Future Research in SNNs

5.1. Challenges and Limitations

SNNs have a number of significant challenges that have delayed their broader adoption. Major limitations include:

- **Accuracy Gaps:** One of the main limitations of deep SNNs is their inability to match the accuracy of conventional machine learning models on standard benchmarks (Pfeiffer & Pfeil, 2018). While there has been progress, the accuracy of SNNs remains lower compared to traditional DNNs. It remains challenging for SNNs to match the accuracy of traditional neural networks on complex tasks, even though SNNs are more energy-efficient by design (Rathi et al., 2023).

- **Learning Capability:** The event-driven nature of SNNs has both advantages and disadvantages. On one hand, spiking neurons are active only when needed, which saves energy. On the other hand, this sparse, event-driven activity limits the network's learning capacity and flexibility (Rathi et al., 2023).

- **Training Difficulties:** Designing and evaluating training algorithms for SNNs is hard due to their discontinuous and asynchronous behavior. This makes it difficult for SNNs to apply the learning methods commonly used in DNNs (Pfeiffer & Pfeil, 2018).

    - Backpropagation in SNNs is unstable and can lead to vanishing gradients or exploding (Rathi et al., 2023; Tavanaei et al., 2019). Lee et al. (2016) presented an innovative method for training deep SNNs via backpropagation, where the backpropagation equations are adapted for LIF neurons. But the method required layer-specific tuning of thresholds and parameters for the training. The approach does not generalize well to hardware yet and assumes availability of detailed spike timing statistics. H. Wu et al. (2021) introduced a novel backpropagation strategy for training SNNs known as ASF-BP, which prevents the vanishing gradient issue and significantly reduces training time compared to other methods. But the proposed method loses temporal precision, which can impact results on neuromorphic datasets. The mathematical model is less effective for LIF neurons in deeper networks. The paper by Y. Wu et al. (2018) presented an innovative algorithm that enhances network accuracy by performing error backpropagation considering both spatial and temporal dimensions. But the approach still requires careful tuning of hyperparameters like decay rate and curve steepness. Although powerful, the algorithm had only been tested on relatively small datasets (MNIST and N-MNIST) and may not yet scale well to larger tasks. Jin et al. (2018) proposed a novel hybrid backpropagation algorithm called HM2-BP, which applied error backpropagation at both the firing rate (macro-level) and spike train (micro-level), as well as their interactions. But the strategy depends on fixed thresholds and Poisson spike encodings, which reduce flexibility. Gradient approximation through decoupling may affect accuracy in complex tasks.

  - Supervised learning in SNNs often adapts traditional methods like surrogate gradient methods to accommodate the non-differentiable behavior of spike events. Neftci et al. (2019) introduced the Surrogate Gradient (SG) methods, which make gradient-based optimization feasible for SNNs. But SGs introduced bias due to approximation and might lead to vanishing gradients with some activation choices. They often lack theoretical grounding and require more empirical validation in complex networks.

  - Unsupervised learning in SNNs typically uses STDP (Legenstein et al., 2005), combined with biologically influenced techniques (Diehl & Cook, 2015; Masquelier & Thorpe, 2007). The biologically inspired SNN introduced by Diehl and Cook (2015) performed unsupervised digit recognition using STDP. Although the proposed model was biologically plausible, it performed slightly lower than some rate-based models. Furthermore, performance depends on careful tuning of inhibitory connections and adaptive thresholds.

  - Indirect learning approaches for SNNs leverage the computational effectiveness of SNNs while preserving the advantages of the pre-training done in an ANN. In this approach, an ANN is first trained using conventional methods before converting it into an SNN. (Rathi et al., 2020; Rueckauer et al., 2017; Rueckauer & Liu, 2018; Sengupta et al., 2019). Sengupta et al. (2019) introduced a data-driven weight normalization strategy called Spike-Norm for ANN-to-SNN conversion that operates without bias terms or batch normalization. However, the method required per-layer tuning of thresholds and specific architecture constraints (such as no biases), which might limit flexibility. Rueckauer et al. (2017) presented a conversion process that allows CNN architectures to be transformed into SNNs. However, the approach had several limitations. For example, deep networks require longer simulation times due to accumulated approximation errors across layers. In larger models, energy savings are less significant compared to smaller networks. Rueckauer and Liu (2018) proposed three variants of the TTFS method to convert ANNs to SNNs. But one of the drawbacks of this strategy is the baseline method can miss important delayed spikes. Dynamic thresholds and clamped ReLU improve accuracy but require retraining or more complex updates. Rathi et al. (2020) presented a hybrid approach to train SNNs, which significantly lowers training complexity. However, the method required constraints like no bias terms and fixed thresholds.

- **Immature Learning Algorithms:** Many learning strategies for SNNs are still at the initial stage (Rathi et al., 2023). They perform well on simple tasks like clustering or basic feature extraction, but they struggle with more complex recognition or decision-making problems.

- **Scalability Issues:** Scalability remains a significant weakness of SNNs. The current frameworks and systems often struggle to handle the demands of large-scale networks. This limits their effectiveness for more complex tasks and leads to potential performance issues (Kudithipudi et al., 2025).

- **Lack of Standard Benchmarks:** The neuromorphic field lacks widely accepted benchmarks to compare different algorithms (Kudithipudi et al., 2025). The diversity of neuromorphic systems complicates the development of universal metrics.

- **Lack of Software Frameworks and Community Involvement:** The neuromorphic field currently lacks open-source and user-friendly software frameworks for designing, training, and deploying spiking networks (Kudithipudi et al., 2025). This increases the difficulty for researchers and developers to experiment with SNNs. The absence of community collaboration also hinders the wider use of SNNs.

- **Adversarial Attacks:** Bagheri et al. (2018) demonstrate that SNNs are vulnerable to white-box adversarial attacks, where small perturbations in the input spike trains can significantly degrade classification accuracy. The paper by Liang et al. (2020) investigates adversarial attacks on SNNs, highlighting difficulties like gradient-related limitations such as gradient incompatibility and vanishing gradients.

5.2. Future Research Directions

SNNs face several challenges that hinder their full potential. Future research should concentrate on overcoming these limitations, which include:

- **Neuron Models and Structural Plasticity:** Future research should focus on neuron models, structural plasticity and other areas to improve learning capacity for SNNs so that it simulates the complicated manner in which the brain works (Rathi et al., 2023). The main goal should be to find the best balance between complexity and the ability for training while designing neuron models and training algorithms together.

- **Expanding SNN Applications:** Current research mostly emphasizes utilizing SNNs for image classification (Rathi et al., 2023). Future research might concentrate on additional tasks, including speech recognition and language modeling.

- **Low-Power Edge Devices:** Future research could concentrate on the widespread adoption of SNNs for low-power edge devices, e.g., IoT sensors and mobile devices (Rathi et al., 2023).

- **Efficient Training Algorithms:** Backpropagation through time requires a lot of memory (Rathi et al., 2023). As a result, future study should aim at designing algorithms for effective training. The goal should be making better use of timing information to speed up training while consuming fewer resources.

- **Improving Batch Normalization:** Batch normalization (Ioffe & Szegedy, 2015) works well for training deep ANNs, but it has been used only in a limited case in SNNs (Rathi et al., 2023). There is a scope to improve this method for SNNs.

- **Handling Event-Based Data:** SNNs excel at handling data from event sensors that change over time (Rathi et al., 2023). Finding more of such activities or finding efficient ways to transform various tasks for SNNs to be handled effectively requires additional research.

- **Resilience to Adversarial Attacks:** The study of making networks robust against adversarial attacks is gaining attention, and SNNs may provide an efficient, energy-saving defense solution (Rathi et al., 2023). Future research in SNNs could aim at enhancing the resilience of these networks to adversarial attacks by exploring improved encoding methods and activation strategies.

- **Scalability:** One of the primary challenges for SNNs lies in their scalability. Future research should emphasize developing scalable frameworks that can handle large networks effectively, which could enhance the performance and applicability of SNNs in more demanding real-world scenarios (Kudithipudi et al., 2025).

- **Distributed and Asynchronous Models:** There is a lack of distributed and asynchronous models for SNNs. Future research should emphasize developing robust distributed architectures and asynchronous learning algorithms. By incorporating strategies from AI's deployment on cloud and heterogeneous systems, scalability and efficiency in large-scale applications can be enhanced. This will accelerate the widespread use of SNNs and neuromorphic systems (Kudithipudi et al., 2025).

- **Community Access:** The progress of SNNs is hindered by limited access to small-scale prototypes for the community. This restriction creates barriers in portability, standardization, and knowledge transfer (Kudithipudi et al., 2025). Future research should concentrate on addressing these limitations.

- **Standardized Benchmarks:** The field of SNNs currently lacks universally accepted benchmarks for comparing various SNN algorithms. This absence hinders the fair evaluation and optimization of different approaches. Future work should aim to create standardized performance metrics and benchmarks to allow more consistent comparisons across diverse neuromorphic systems. (Kudithipudi et al., 2025).

- **Integrating SNNs with Deep Learning Models:** Future research should explore integrating SNNs with traditional deep learning models to combine the strengths of both strategies, which could result in more efficient and flexible network designs (Sanaullah et al., 2023).

- **Spike Encoding Schemes:** Future research should prioritize the investigation of various spike encoding schemes to improve how information is represented within SNNs. Designing more effective spike coding techniques could boost SNN performance while minimizing computational costs (Sanaullah et al., 2023).

6. Conclusion

Spiking Neural Networks hold considerable promise by mimicking the brain's biological processes, offering advantages in energy efficiency and temporal information processing (Sanaullah et al., 2023). However, despite these benefits, several challenges remain that have limited their widespread adoption. These include issues with the difficulty in training due to the event-driven nature of SNNs, accuracy gaps compared to traditional DNNs, scalability and the absence of standardized benchmarks for evaluating various algorithms. While progress has been made with biologically inspired learning methods like STDP (Diehl & Cook, 2015) and hybrid models combining SNNs and traditional ANNs (Rathi et al., 2020; Rueckauer et al., 2017; Rueckauer & Liu, 2018; Sengupta et al., 2019), the full potential of SNNs is still not fully realized.

This survey offers a structured overview of SNNs and their learning strategies, though it has certain limitations. The focus of this paper was primarily on the learning and training aspects of SNNs. Moreover, while comparisons with other competing surveys were made, future research should consider additional factors like hardware implementation and real-world applications of SNNs, which are not fully explored here. For example, Bouvier et al. (2019) focused on SNN hardware architecture, Yamazaki et al. (2022) covered the applications of SNNs, Niu et al. (2023) provided the current research progress of SNNs in image classification tasks, Rathi et al. (2023) offered a comprehensive overview of the integration of neuromorphic hardware with SNN algorithms and Kudithipudi et al. (2025) emphasized the challenges of neuromorphic hardware/software co-design. In contrast, this survey is unique in its comprehensive concentration on the SNN's foundations and learning strategies, which are essential for the progress of the field. Other surveys may have placed greater emphasis on specific areas, while this paper concentrated on providing a solid technical foundation for the implementation of SNNs.

The future research should focus on overcoming the training, scalability and performance challenges mentioned by designing distributed, asynchronous models, improving training algorithms, and integrating SNNs with deep learning approaches. It will also be important to optimize spike encoding strategies, establish universally accepted benchmarks and enhance the robustness of SNNs against adversarial attacks. Addressing these challenges and the issues highlighted in other surveys is necessary to speed up the use of SNNs in real-world situations, which will allow energy-efficient, biologically inspired solutions.

7. References

Bagheri, A., Simeone, O., & Rajendran, B. (2018). *Adversarial Training for Probabilistic Spiking Neural Networks* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1802.08567

Bohte, S. M., Joost N Kok, & Han La Poutré. (2000). SpikeProp: Backpropagation for Networks of Spiking Neurons. *In ESANN*, *(Vol. 48*, 419-424). Retrieved from https://api.semanticscholar.org/CorpusID:14069916

Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., & Beigne, E. (2019). Spiking Neural Networks Hardware Implementations and Challenges: A Survey. *ACM Journal on Emerging Technologies in Computing Systems*, *15*(2), 1–35. https://doi.org/10.1145/3304103

Buzsáki, G. (2006). *Rhythms of the Brain*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780195301069.001.0001

Cruz-Albrecht, J. M., Yung, M. W., & Srinivasa, N. (2012). Energy-Efficient Neuron, Synapse and STDP Integrated Circuits. *IEEE Transactions on Biomedical Circuits and Systems*, *6*(3), 246–256. https://doi.org/10.1109/TBCAS.2011.2174152

Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, *9*, 99. https://doi.org/10.3389/fncom.2015.00099

Ghosh-Dastidar, S., & Adeli, H. (2009). SPIKING NEURAL NETWORKS. *International Journal of Neural Systems*, *19*(04), 295–308. https://doi.org/10.1142/S0129065709002002

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *117*(4), 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764

Hunsberger, E., & Eliasmith, C. (2015). *Spiking Deep Networks with LIF Neurons*. arXiv preprint arXiv: 1510.08829. https://doi.org/10.48550/ARXIV.1510.08829

Iaboni, C., & Abichandani, P. (2024). Event-based Spiking Neural Networks for Object Detection: Datasets, Architectures, Learning Rules, and Implementation. *IEEE Access*, *12*, 180532–180596. https://doi.org/10.1109/ACCESS.2024.3479968

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, 448–456. https://doi.org/10.48550/ARXIV.1502.03167

Jin, Y., Zhang, W., & Li, P. (2018). Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks. *Advances in Neural Information Processing Systems*, *31*. https://doi.org/10.48550/ARXIV.1805.07866

Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization* (Version 9). arXiv preprint arXiv:1412.6980. https://doi.org/10.48550/ARXIV.1412.6980

Kudithipudi, D., Schuman, C., Vineyard, C. M., Pandit, T., Merkel, C., Kubendran, R., Aimone, J. B., Orchard, G., Mayr, C., Benosman, R., Hays, J., Young, C., Bartolozzi, C., Majumdar, A., Cardwell, S. G., Payvand, M., Buckley, S., Kulkarni, S., Gonzalez, H. A., … Furber, S. (2025). Neuromorphic computing at scale. *Nature, 637*(8047), 801–812. https://doi.org/10.1038/s41586-024-08253-8

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training Deep Spiking Neural Networks Using Backpropagation. *Frontiers in Neuroscience, 10*, 508. https://doi.org/10.3389/fnins.2016.00508

Legenstein, R., Naeger, C., & Maass, W. (2005). What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? *Neural Computation*, *17*(11), 2337–2382. https://doi.org/10.1162/0899766054796888

Liang, L., Hu, X., Deng, L., Wu, Y., Li, G., Ding, Y., Li, P., & Xie, Y. (2020). *Exploring Adversarial Attack in Spiking Neural Networks with Spike-Compatible Gradient* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2001.01587

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7

Markram, H., Gerstner, W., & Sjöström, P. J. (2012). Spike-Timing-Dependent Plasticity: A Comprehensive Overview. *Frontiers in Synaptic Neuroscience*, *4*, 2. https://doi.org/10.3389/fnsyn.2012.00002

Masquelier, T., & Thorpe, S. J. (2007). Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity. *PLoS Computational Biology*, *3*(2), e31. https://doi.org/10.1371/journal.pcbi.0030031

McKennoch, S., Dingding Liu, & Bushnell, L. G. (2006). Fast Modifications of the SpikeProp Algorithm. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 3970–3977. https://doi.org/10.1109/IJCNN.2006.246918

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine, 36*(6), 51–63. https://doi.org/10.1109/MSP.2019.2931595

Niu, L.-Y., Wei, Y., Liu, W.-B., Long, J.-Y., & Xue, T. (2023). Research Progress of spiking neural network in image classification: A review. *Applied Intelligence, 53*(16), 19466–19490. https://doi.org/10.1007/s10489-023-04553-0

Pfeiffer, M., & Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in Neuroscience, 12*, 774. https://doi.org/10.3389/fnins.2018.00774

Pietrzak, P., Szczęsny, S., Huderek, D., & Przyborowski, Ł. (2023). Overview of Spiking Neural Network Learning Approaches and Their Computational Complexities. *Sensors, 23*(6), 3037. https://doi.org/10.3390/s23063037

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*(12), 1–49. https://doi.org/10.1145/3571155

Rathi, N., Srinivasan, G., Panda, P., & Roy, K. (2020). *Enabling Deep Spiking Neural Networks with Hybrid Conversion and Spike Timing Dependent Backpropagation* (Version 1). arXiv preprint arXiv:2005.01807. https://doi.org/10.48550/ARXIV.2005.01807

Rueckauer, B., & Liu, S.-C. (2018). Conversion of analog to spiking neural networks using sparse temporal coding. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. https://doi.org/10.1109/ISCAS.2018.8351295

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image

Classification. *Frontiers in Neuroscience*, *11*, 682. https://doi.org/10.3389/fnins.2017.00682

Ruf, B., & Schmitt, M. (1997). Learning temporally encoded patterns in networks of spiking neurons. *Neural Processing Letters*, *5*(1), 9–18. https://doi.org/10.1023/A:1009697008681

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. https://doi.org/10.3389/fncom.2023.1215824

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in Neuroscience*, *13*, 95. https://doi.org/10.3389/fnins.2019.00095

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (Version 6). arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/ARXIV.1409.1556

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926. https://doi.org/10.1038/78829

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.48550/ARXIV.1512.00567

Tan, C., Šarlija, M., & Kasabov, N. (2020). Spiking Neural Networks: Background, Recent Development and the NeuCube Architecture. *Neural Processing Letters*, *52*(2), 1675–1701. https://doi.org/10.1007/s11063-020-10322-8

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, *111*, 47–63. https://doi.org/10.1016/j.neunet.2018.12.002

Thorpe, S., & Gautrais, J. (1998). Rank Order Coding. In J. M. Bower (Ed.), *Computational Neuroscience* (pp. 113–118). Springer US. https://doi.org/10.1007/978-1-4615-4831-7_19

Wang, X., Lin, X., & Dang, X. (2020). Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, *125*, 258–280. https://doi.org/10.1016/j.neunet.2020.02.011

Wu, H., Zhang, Y., Weng, W., Zhang, Y., Xiong, Z., Zha, Z.-J., Sun, X., & Wu, F. (2021). Training Spiking Neural Networks with Accumulated Spiking Flow. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 10320–10328. https://doi.org/10.1609/aaai.v35i12.17236

Wu, J., Chua, Y., Zhang, M., Yang, Q., Li, G., & Li, H. (2019). Deep Spiking Neural Network with Spike Count based Learning Rule. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–6. https://doi.org/10.1109/IJCNN.2019.8852380

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2018). Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Frontiers in Neuroscience*, *12*, 331. https://doi.org/10.3389/fnins.2018.00331

Xu, Y., Zeng, X., & Zhong, S. (2013). A New Supervised Learning Algorithm for Spiking Neurons. *Neural Computation*, *25*(6), 1472–1511. https://doi.org/10.1162/NECO_a_00450

Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., & Le, N. (2022). Spiking Neural Networks and Their Applications: A Review. *Brain Sciences*, *12*(7), 863. https://doi.org/10.3390/brainsci12070863

Zenke, F., & Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, *30*(6), 1514–1541. https://doi.org/10.1162/neco_a_01086

Zhang, L., Zhou, S., Zhi, T., Du, Z., & Chen, Y. (2019). TDSNN: From Deep Neural Networks to Deep Spike Neural Networks with Temporal-Coding. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 1319–1326. https://doi.org/10.1609/aaai.v33i01.33011319

Zhang, W., & Li, P. (2020). *Temporal Spike Sequence Learning via Backpropagation for Deep Spiking Neural Networks* (Version 4). arXiv. https://doi.org/10.48550/ARXIV.2002.10085