Journal of
Low Power Electronics and Applications

MDPI

Review

A Review of Algorithms and Hardware Implementations for Spiking Neural Networks

Duy-Anh Nguyen [1,2], Xuan-Tu Tran [1,*] and Francesca Iacopi [3]

[1] VNU Information Technology Institute, Vietnam National University, Hanoi 123106, Vietnam; danguyen@vnu.edu.vn
[2] VNU University of Engineering and Technology, Vietnam National University, Hanoi 123106, Vietnam
[3] Faculty of Engineering & IT, University of Technology Sydney, Broadway 2007, Australia; francesca.iacopi@uts.edu.au
* Correspondence: tutx@vnu.edu.vn

Abstract: Deep Learning (DL) has contributed to the success of many applications in recent years. The applications range from simple ones such as recognizing tiny images or simple speech patterns to ones with a high level of complexity such as playing the game of Go. However, this superior performance comes at a high computational cost, which made porting DL applications to conventional hardware platforms a challenging task. Many approaches have been investigated, and Spiking Neural Network (SNN) is one of the promising candidates. SNN is the third generation of Artificial Neural Networks (ANNs), where each neuron in the network uses discrete spikes to communicate in an event-based manner. SNNs have the potential advantage of achieving better energy efficiency than their ANN counterparts. While generally there will be a loss of accuracy on SNN models, new algorithms have helped to close the accuracy gap. For hardware implementations, SNNs have attracted much attention in the neuromorphic hardware research community. In this work, we review the basic background of SNNs, the current state and challenges of the training algorithms for SNNs and the current implementations of SNNs on various hardware platforms.

Keywords: spiking neural networks; deep neural networks; deep learning; FPGA; digital design

Citation: Nguyen, D.-A.; Tran, X.-T.; Iacopi, F. A Review of Algorithms and Hardware Implementations for Spiking Neural Networks. *J. Low Power Electron. Appl.* **2021**, *11*, 23. https://doi.org/10.3390/jlpea11020023

Academic Editors: Ivan Miro-Panades, Andrea Calimera, Koushik Chakraborty and Amit Kumar Singh

Received: 20 April 2021
Accepted: 19 May 2021
Published: 24 May 2021

1. Introduction

Artificial Neural Networks (ANNs) have pushed the development of machine learning algorithms for the past few decades. ANNs are loosely inspired from the biological brain and are built with computing elements called *neurons*. These neurons receive a set of weighted inputs from neurons of the previous layer, have continuous activation values and utilize differentiable non-linear activation functions. The neurons are grouped as a layer, and many layers are stacked to create a very deep network topology. The differentiable nature of such activation functions allows the usage of gradient-based optimization methods such as backpropagation to train them, i.e., fine-tuning the parameters of the networks to fit with a set of desired outputs. Recent advances in computing power with the GPU platform, coupled with the availability of large labeled datasets, have made the training of such very deep networks possible. The research field has been termed *Deep Learning* (DL), and very deep networks with many layers fall under the umbrella term of *Deep Neural Network* (DNN). DNN has been successfully applied to many areas, including image recognition [1–3], object detection [4], speech recognition [5] or playing the game of Go [6].

Although DNNs are brain-inspired, there are some fundamental differences between the processing of DNNs and the way the human brain works. The most important difference is the way information is represented and communicated between primitive computing elements. DNNs represent inputs as continuous-valued activations, and these values are transmitted and accumulated as weighted inputs for the downstream neurons. On the other hand, biological neurons communicate with each other through trains of

action potentials called *spikes*. Each pair of neurons forms a connection called a *synapse*. The spikes are sparse in time, and the information is represented with either the spike timing or the spike rates over a specific time window. Another difference is the way the learning process happens in the biological brain versus the training process of DNNs. The gradient-based learning method is not bio-plausible, as the synapse strength modification process between biological neurons is observed to depend on the relative timing of input and output spike. The information required for such learning rules is only available locally between a connected pair of neurons, and it does not depend on other neurons in the network. On the other hand, the gradient-based learning method is a method to optimize a single loss function, which depends on every connection between layers in the network.

The aforementioned observations have led to the birth of *Spiking Neural Networks* (SNNs)—the third generation of Artificial Neural Networks. SNNs are more closely related to the human brains, as the neurons communicate with each other through spikes, and the weighted connection between pair of neurons are modifiable through some forms of Spike Timing Dependent Plasticity (STDP). Compared to its DNN counterpart, SNN offers some promising advantages. Firstly, Maass [7] proved that the computing capability of SNN is at least equal to that of ANN while requiring fewer computing elements. Secondly, the spikes are infrequent in time, and the communication is event-driven. This leads to a reduction in power consumption. Lastly, SNN can capture the temporal characteristics of data. The spike timing plays an important role in many different coding strategies of inputs. With those advantages, SNN has been applied to many areas such as visual processing [8–10], speech recognition [11,12] and medical diagnosis [13,14]. In recent years, the deep network topology with the multi-layer structure of ANN and the bio-inspired spiking mechanism has been extensively investigated, and these kinds of networks are called *Deep Spiking Neural Networks* (DSNN).

As the network size of DNNs and SNNs grows, their computational complexity has made their evaluation on traditional von Neumann computer architecture very time-consuming and energy inefficient. The VLSI research community has made considerable efforts to develop dedicated hardware architecture to accelerate the testing of DNN and SNN algorithms. Since these algorithms mirror the biological computing process in the brain, it is only natural that those hardware architectures are also taking inspiration from the brain structure. These systems are referred to as *neuromorphic computing*. Neuromorphic computing is expected to lead ultimately to better energy efficiency compared to traditional computer architecture, due to the event-driven nature of the computation. However, it is still not clear whether the current hardware implementations of SNNs are efficient.

In this work, we explain the basics of SNNs in Section 2. We classify the basic training algorithms for SNNs in Section 3. Hardware Implementations are highlighted in Section 4. Future trends of research are presented and discussed in Section 5. Finally, Section 6 concludes the paper.

2. Fundamentals of Spiking Neural Networks

In this section, dynamics of spiking neurons and synapses as well as some popular encoding strategies are presented.

2.1. Neuron Models

Despite many research efforts over the past couple of decades, understanding the human brain's neural structure remains one of the most challenging tasks. Based on the observed dynamics of biological neurons in the human brain, there have been many proposed neuron models. Hodgkin and Huxley [15] were among the first to propose a model with extensive biological detail; however, it comes with a high computational cost. Izhikevich [16] proposed a neuron model that can capture diverse dynamic behaviors of a biological neuron while keeping a moderate computational efficiency. The Leaky Integrate and Fire (LIF) and the simplified version Integrate and Fire (IF) are among the most popular models, as they capture the intuitive property of accumulating charge on the neuron's

membrane, with a constant leaky charge and a clear threshold. Figure 1 depicts a spiking neuron.

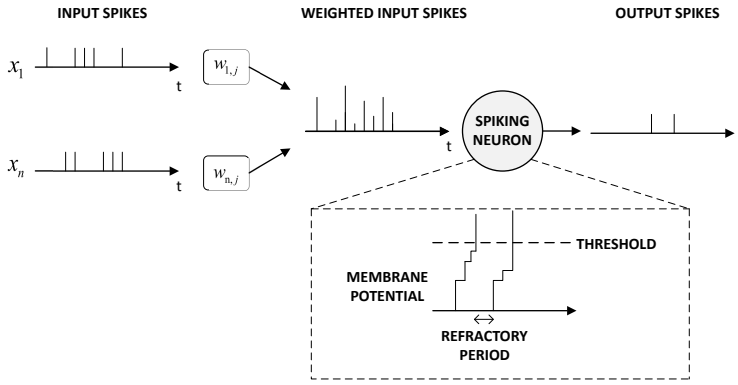

Figure 1. Schematic representation of a spiking neuron. The neuron receives many input spikes from the previous layers' neurons. Each input spike is modulated with a weight, producing weighted synaptic inputs. Those inputs are accumulated as the neuron membrane potential. If the membrane potential crosses a certain threshold, the neuron will emit a spike to the downstream layer and the membrane potential will be reset. A refractory period, where the neuron remains inactive (i.e., does not further integrate any incoming spikes) after firing, may be implemented in some models.

The LIF is easily modeled and implemented in hardware [17–20] and has been applied in a wide range of applications. The temporal dynamics of a LIF neuron are described in the following.

Let $x_{i,j}(t-1)$ represent the input spike from the neuron i in previous layer to neuron j in the current layer. A neuron is characterized by its membrane potential V_m, potential threshold V_{th} and refractory period T_R. The membrane potential at each time step t can be described by

$$V_m(t) = V_m(t-1) + \sum_i w_{i,j} \times x_{i,j}(t-1) - \lambda. \tag{1}$$

Whenever neuron i receives an input spike from a previously connected neuron j, it will integrate the synapse weight corresponding to this connection $w_{i,j}$ into the membrane potential. Due to leakage, its membrane potential will be decreased by a constant amount λ between two input spikes. If the membrane potential crosses the threshold, the neuron will fire and reset its membrane potential. The output spike is sent to all the neurons in the next layer. The refractory period represents the neuron's period of inactivity after firing, i.e., the neuron will not fire in this period even if its membrane potential is above the threshold. The output spike of a neuron is given by

$$x(t) = \begin{cases} 1 & \text{if } V_m(t) > V_{th} \text{ and } t - t_{spike} > T_R \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where t_{spike} is the time step at which the neuron fired. After firing, the neuron will reset its membrane potential. There are two popular reset mechanisms: either reset to a constant value V_{reset} or subtract a constant value from the current membrane potential.

Even though the LIF neuron model is currently the most popular when implemented in a hardware system, simulating other neuron models with more bio-realistic features in dedicated hardware platform is also gaining interests lately, such as the Hodgkin and Huxley model [21–24] or the Izhikevich model [25–27]. The models' dynamics are often defined with a set of differential equations, and they present a challenge to faithfully reproduce the complex dynamics of such models on hardware platform such as FPGA. Usually, a reduced and hardware-friendly version of those models are used such as the Piecewise Linear Models (PWLs) [26] or base-2 exponential functions [21]. The approximated functions help to estimate the continuous functions in the set of differential equations. Model parameters

are often fixed and simplified through some optimization methods [21,25]. After getting a simplified version of the neuron model, a standard numerical method such as the Euler method is used to get a discretized model for hardware platform.

2.2. Synapse Models

Each neuron forms a connection called *synapse* with downstream neurons in the next layer. Every time a neuron fires, the output spike will stimulate the membrane potential of downstream neurons. There are two kinds of synaptic behaviors, namely the excitatory synapse and inhibitory synapse. In an excitatory connection, an input spike increases the post-synaptic neurons' potential, while decreasing the potential in an inhibitory connection. The amount of change is determined by the synapse strength, i.e., the weight of the connection between two neurons. In a formal ANN network, the weight is a scalar value that is predetermined through an off-line training process and remains constant during the inference phase. SNNs take a more bio-plausible approach. In SNNs, the synapses experience the phenomenon of plasticity, where the synapse strength is adjustable and found to depend on the relative timing between pre- and post-synaptic spikes of a neuron.

2.3. Encoding Information with Binary Input Spikes in SNNs

In ANNs, inputs are usually represented as analog values, such as the pixel values in image recognition applications, voice data for speech recognition applications, etc. However, in SNNs input information must be presented to each neuron in the form of binary spike trains. Hence, there is a need to encode complex analog values to binary form, and how to best represent such information is still a topic of discussion [28–30].

The majority of the SNN architectures use the rate-coding method to represent information with binary spikes. The mean firing rate of input spikes over a timing window represents the analog values. Let $x \in [0, 1]$ denote the analog values that are represented with the spike trains. At each time step t, the binary spike train $X(t)$ is given by

$$X(t) = \begin{cases} 1 & \text{if } x < u \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $u \sim U(0, 1)$ is a uniformly distributed random number generated in each time step. The probability of observing a spike in a total of T time steps $P(T)$ is proportional to the actual analog value x. The timing between two consecutive spikes is random, but the overall frequency is preserved. This leads to the two major drawbacks of the rate coding method. First, it cannot capture the temporal characteristics of the data, as the precise timing of each spike is irrelevant. Second, during a large timing window, the spike trains may become very sparse or very dense, depending on how the random numbers are generated.

Neuroscience research has shown that much information in the human brain is encoded with temporal characteristics, i.e., timing information plays a crucial role. Many time coding methods have been proposed, among those the two most popular methods are Time To First Spike (TTFS) and Inter-Spike Interval (ISI). In TTFS, the analog value x is proportional to the arrival time of input spikes, and each analog value can be represented with only one spike. In ISI, the values are coded with the precise timing between two consecutive spikes. Figure 2 shows the three mentioned coding schemes.

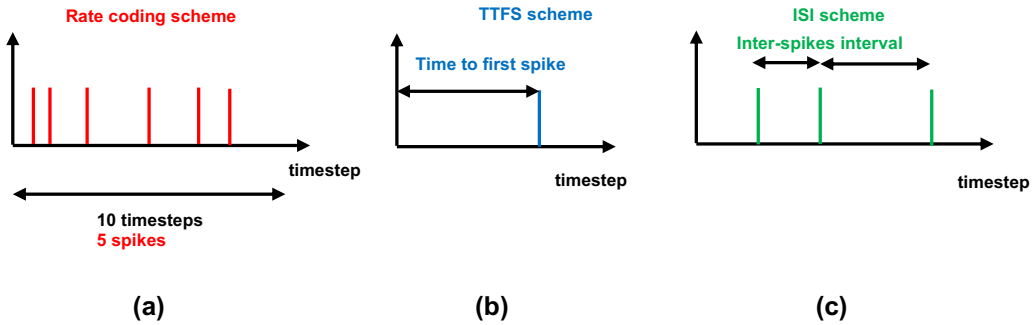

Figure 2. Several coding schemes for encoding analog values to spike trains. (**a**) The rate-coding scheme. The analog values are represented as the rate at which input spikes are produced. In the example, there are five input spikes in a span of 10 timesteps. Hence, the analog value of $5 \div 10 = 0.5$ is encoded. (**b**) The TTFS scheme. In this encoding scheme, each analog value could be represented with only one input spike. The analog value to be encoded is inversely proportional to the delay of the input spike, i.e., a higher analog value will produce an earlier input spike. (**c**) The ISI scheme. In this scheme, the precise inter-spike timings between two consecutive spikes are used to encode the analog input value.

3. Learning Rules in Spiking Neural Networks

3.1. Unsupervised Learning with STDP

Unsupervised learning is the process of learning without pre-existing labels. In SNNs, most unsupervised learning algorithms use some forms of STDP as a part of the learning procedure [31,32]. STDP is a Hebbian learning rule which has a very intuitive explanation. The synapse's strength between two neurons depends on the relative timings of the pre- and post-synaptic spikes. The pre-synaptic spikes are the incoming spikes from the previous layer to the neuron. The post-synaptic spikes are the output spikes from the neuron. If a pre-synaptic neuron fires briefly (in a 10 ms time frame) before the post-synaptic neuron, then the connection between them is strengthened, i.e., the weight is increased. This phenomenon is called long-term potentiation (LTP). If the pre-synaptic neuron fires briefly after the post-synaptic neuron, then the connection between them is weakened, i.e., the weight is decreased. This is called long-term depression (LDP). Equation (4) depicts the experimentally observed STDP learning rule [33]:

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} \leq 0 \text{ and } A > 0 \\ Be^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} \geq 0 \text{ and } B < 0 \end{cases} \tag{4}$$

where Δw is the weight change, A and B are constant parameters, τ is the timing window constant (usually $\approx$ 10 ms [34]) and t_{pre} and t_{post} are the absolute timing of the pre- and post-synaptic spikes with a given timescale, respectively. The first case indicates LTP while the second case indicates LDP. However, when applied to most machine learning problems, this form of STDP is rarely used, and usually, a simplified form of STDP is preferred. One major drawback of STDP is that it does not perform well in terms of accuracy when applied to multi-layer networks. This is due to the nature of STDP, where the learning rule strictly depends on the connection between a layer and its preceding layer while lacking coordination with other parts of the network. Figure 3 shows a simplified STDP learning rule.

3.2. Supervised Learning with Backpropagation

Almost all supervised learning rules in SNNs are based on some gradient-based optimization methods. The learning process tries to minimize the error between the desired output and output spike trains, given a specific input. Supervised learning in SNNs has the advantage of reaching accuracies equivalent to the ones of formal ANNs. However, the main factor that limits the development of efficient backpropagation with spiking neurons is that, since spikes are discrete in time, their activation function is not derivable. The key

enabling backpropagation in multi-layer SNNs is to find an approximate, real-valued and differentiable surrogate to the activation function of spiking neurons.

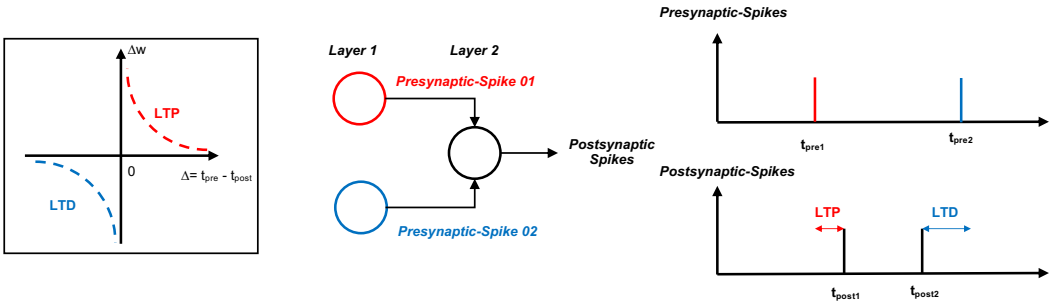

Figure 3. Unsupervised learning with STDP. (**a**) A graph showing the relationship between the change in synapse strength Δw and the relative timing between the pre- and post-synaptic spikes Δt. The synapse strength decreases or increases exponentially with the relative timing. (**b**) A diagram showing the interaction between neurons in two successive layers (layer 1 and layer 2 in the figure). The pre-synaptic spike from neuron 1 (in red) comes before the post-synaptic spikes, hence the synapse strength increases (LTP). In the case of neuron 2 (in blue), the post-synaptic spike happens after the pre-synaptic spike from neuron 2. In this case, the synapse strength decreases (LTD).

The earliest attempts to train SNNs by backpropagating errors is SpikeProp [35] and its variants. SpikeProp takes into account the spike timing in the cost function and can classify the XOR problem using a temporally coded spike and three-layer architecture. However, it does not scale well with modern large-scale deep learning problems, partly due to being too computationally expensive. In [36], the authors treated the membrane potentials as a differentiable signal, and discontinuities at spikes are treated as noise and handled with a low-pass filter before backpropagation. In [37], the timing of the first spike for each neuron is used as its activation value during training, thus the network input–output function is differentiable almost everywhere. The network becomes very sparse while still being able to process complex temporal information of input spikes. A hybrid approach that combines backpropagation in both spatial domain and time domain is proposed in [38]. This opens up the possibility to investigate high-performance SNN for future event-driven datasets with complex spatiotemporal statistics.

3.3. Conversion of SNN from DNN

To avoid the need for complex and dedicated training procedures for SNNs, direct conversion of trained ANNs to SNNs has been extensively investigated. The goal is to train a known ANN network first, and then adapt its weights and parameters to its equivalent SNN counterpart, as depicted in Figure 4.

The main advantages of the conversion approach lie in two aspects. The first one is that we could leverage highly efficient training techniques developed for ANNs, and many state-of-the-art deep networks for classification tasks can be successfully converted to SNNs [39,40]. Originally, the conversion required some modifications on the network topology of the formal network [41]. Recently, it is possible to directly train DNNs without consideration for the conversion process later, and the conversion process only involves some simple transformations, adding a negligible training overhead [42,43]. The second advantage is that the conversion approach sets benchmark records in terms of accuracy for SNNs, even for a more challenging dataset, such as CIFAR-10 or ImageNet [40], and the accuracy loss compared to its formal ANNs is small.

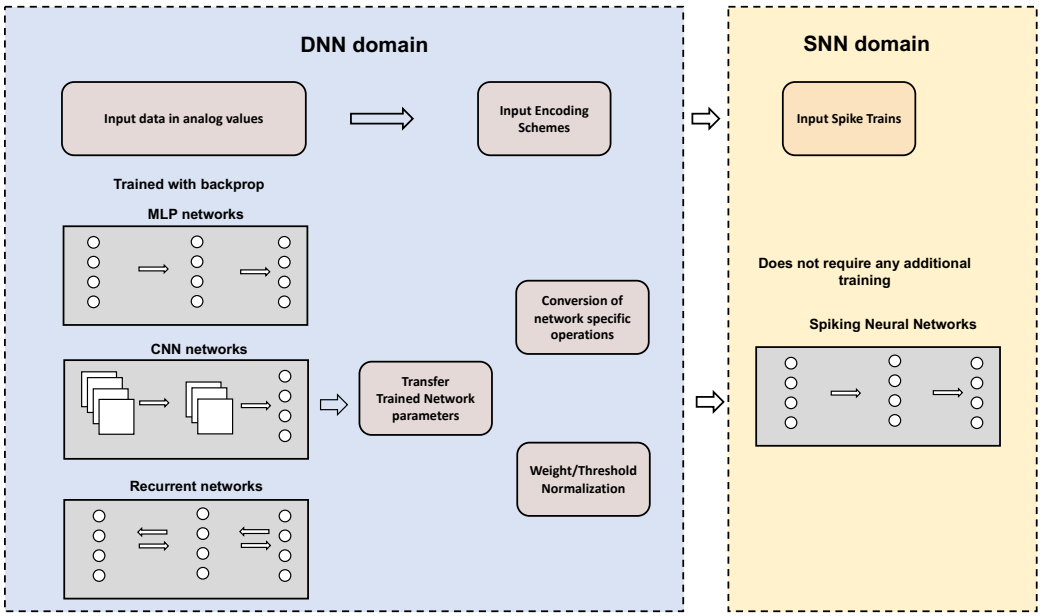

Figure 4. The conversion flow from DNNs to SNNs. Many different types of pre-trained networks from the DNN domain, such as Convolutional Neural Network (CNN), Multi-layer Perceptrons (MLP) or Recurrent Neural Network (RNN), can be converted to the SNN domain. The network parameters are directly transferred to the SNN model. Usually, to preserve accuracy, a weight/threshold normalization scheme is required. Many different network-specific operations may need to be modified, such as the max-pooling operations and the soft-max layer of CNN, or the gated operations of RNN. The input data with analog values in the DNN domain also need to be converted to spike trains with an input encoding scheme.

However, the conversion approaches still have some flaws. The first is that not all ANN operations can be converted to SNN operations, namely the activation functions and the max-pooling operations. The second is that the conversion approach used is usually the rate-coding method, which typically requires many input spikes and degrades the power efficiency once implemented in the hardware platform.

4. Hardware Implementations of SNNs

The hardware implementations of SNNs could be broadly classified into two large categories. The first one is the large-scale neuromorphic accelerator, which is aimed at supporting large-scale applications such as brain simulation or simulation of SNNs with very large topologies. These kinds of computing platforms should focus on providing high throughput and scalable architecture with many cores on-chip and multi-chip configurations. On the other side of the spectrum is the embedded neuromorphic platform, which is targeted to handling SNNs for edge applications. These small-scale platforms should focus on optimizing the power consumption and the accuracy of the SNN applications when mapped to those platforms with tight constraints. We review some notable works in those two categories.

4.1. Large-Scale Neuromorphic Accelerator

4.1.1. General Strategy

Figure 5 shows a general architecture for large-scale, scalable neuromorphic hardware. A large-scale SNN network is characterized by a large amount of memory access for each neuron updates, which could cause a memory bottleneck in the traditional von Neumann architecture. To mitigate such bottlenecks, the general idea is to have many neuromorphic cores which operate in parallel. Each core has dedicated neuron update logic and synapse memory, hence neuron state and synapse weight updates can be handled locally. Different cores communicate with a network-on-chip, usually with Address-Event-Representation

(AER) protocol. In the AER protocol, each neuron has a unique address, which is sent to the AER bus whenever the neuron fires. The firing time is encoded in real-time.

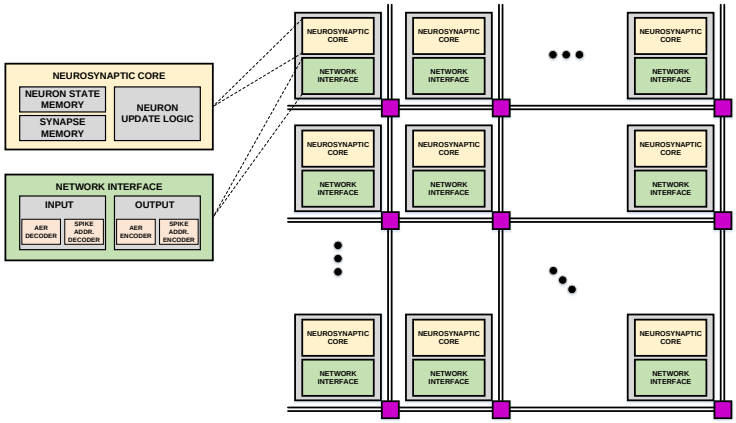

Figure 5. General strategy for the large-scale implementation of neuromorphic hardware.

4.1.2. Comparison of Large-Scale Neuromorphic Accelerator

In this section, we report the four largest and most notable systems in the field of neuromorphic hardware: SpiNNaker [44], TrueNorth [45], Neurogrid[46] and Loihi [47].

SpiNNaker is a very large-scale digital system designed to simulate large spiking neural networks to emulate the human brain in real-time. Each chip in the SpiNNaker system consists of 18 processor cores, and 48 chips are grouped into one board. Each processor core contains both local memory and shared memory between cores. The cores communicate through a custom network-on-chip that specializes in routing small packets with high efficiency. The chips across boards communicate with direct connections. To further improve the scalability, larger configurations could be built by connecting 10 boards. The largest configuration contains over one million cores. SpiNNaker is very configurable; however, it offers lower energy efficiency when simulating complex neurons and synapses behaviors.

Neurogrid is a large SNN emulator which is designed to support complex neural behavior in real-time. Neurogrid employs a mixed-signal subthreshold circuit to model the neurons with bio-realistic mechanisms. Each Neurogrid board consists of 16 cores, connected with a network-on-chip with a tree topology, and supports multicast. Each chip consists of a 2D array of 256×256 neurons. The largest configuration supports up to one million neurons and billions of sparse synaptic connectivity.

TrueNorth is a 28-nm digital CMOS neuromorphic chip developed by IBM in 2014, to evaluate very large-scale SNN operations while keeping an ultra-low power consumption level. One TrueNorth chip has 4096 synaptic cores, with local SRAM memories to store neurons and synapses states. The digital neuron circuit could emulate an extended version of leaky-integrate-and-fire dynamics, and the operations of each neuron are time-multiplexed; hence, one logic circuit could handle the operations of 256 neurons. The synapse memory is designed to support limited weight values and to store the connectivity between neurons. A custom asynchronous network-on-chip is utilized to connect any neurons in any core to any neurons in different cores. The routing infrastructure also permits the integration of multiple TrueNorth chips in a single board. TrueNorth can reach a peak throughput of 46 billion synaptic operations per second (SOPS), while only consuming 26 pico Joule per synaptic event.

Loihi is a recent neuromorphic chip demonstrated by Intel in the 14-nm FinFET process in 2018. It features a platform to study and simulate many different SNN topologies, with various kinds of adaptive neuron models and many online-learning algorithms. One Loihi chip consists of 128 neural cores, with 1024 spiking neurons in each core. Each chip also has three embedded x86 processors and local SRAMs to store connectivities as well as synaptic weights. The spike communications between neurons are handled with an asynchronous network-on-chip, which allows the connection of up to 4096 on-chip cores and 16,384 chips.

One Loihi chip can reach up to 30 billion SOPS throughput while consumes 15 pico Joule per synaptic operation.

We have put together a table of comparison to describe various characteristics of the large-scale accelerators (Table 1).

Table 1. Large-scale accelerator comparison.

Processor	SpiNNaker [44]	Neurogrid [46]	TrueNorth [45]	Loihi [47]
Implementation	Digital	Analog	Digital	Digital
Technology	130 nm	180 nm	28 nm	14 nm
Weight Resolution	8b–32b	13b	1b–4b	1b–64b
Online learning	Yes	No	No	Yes
Neurons per cores	1000	65,000	256	1024
Cores per chip	16	1	4096	128
Energy/SOPS (pJ)	27,000	941	26	15

4.2. Low-Power SNN Accelerator

In this section, we report the status of low-power SNN accelerators which focus on image recognition applications. We choose this kind of applications as it offers a broad spectrum of practical applications such as autonomous cars, robots and drones. Most of the major low-power SNN accelerators are benchmarked with the MNIST dataset, even though it is mainly used to benchmark formal ANN networks. The MNIST dataset allows multiple SNN topologies to be trained with various online learning rules, while still reaching high accuracies. Hence, we focus on neuromorphic hardware evaluation on the MNIST dataset.

Frenkel et al. [48] introduced ODIN, a neuromorphic chip that supports the evaluation of shallow SNNs with a form of on-chip spike-driven synaptic plasticity (Figure 6). The digital neuronal circuits could support both LIF models and Izhikevich models. One physical neuronal circuit is time-multiplexed to emulate 256 neurons dynamic in a sequential fashion. Local SRAM arrays are utilized to store both the neuron parameters and the synaptic weights. With a pre-processed MNIST dataset with lower resolution, the authors reached a peak of 84.5% accuracy with 15 nJ per inference. The authors demonstrated the ability to apply online learning on an ultra-low-power platform, a high level of energy efficiency, but the trade off is the classification accuracy with a much simpler MNIST dataset.

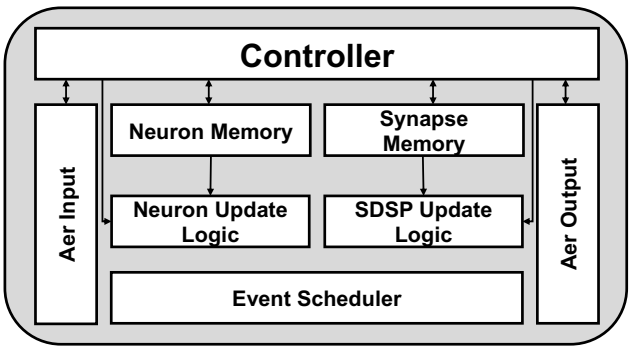

Figure 6. ODIN chip block-level diagram by Frenkel et al. [48]. The chip supports a form of online learning based on Spike-driven synaptic plasticity (SDSP). Local neuron memory and synapses memory is used to time-multiplex the dynamics of 256 neurons, using only a single neuron logic. The inputs and outputs of the chip use AER protocol to handle event-driven, asynchronous outputs. The neuron update logic supports both simple LIF and Izhikevich [16] neuron models.

Yin et al. [49] presented a novel modified LIF neuron model as illustrated in Figure 7, modeled with binary activations, which enabled the offline backpropagation training methods. The work applies the straight-through-estimator method to estimate the gradients. The novel contributions include a modified LIF model, which enables the supervised

training methods. The hardware was implemented in a CMOS 28 nm technology, and various SNNs with fully connected topologies are benchmarked. The work was able to reach high accuracy (over 99% for MNIST) while consuming 51.4–773 nJ per inference. Due to the efficient training methods, this work has reported one of the highest MNIST accuracy results, while keeping the energy efficiency relatively high. However, the main shortcoming of this work is that the hardware architecture is fixed to a shallow network of three layers.

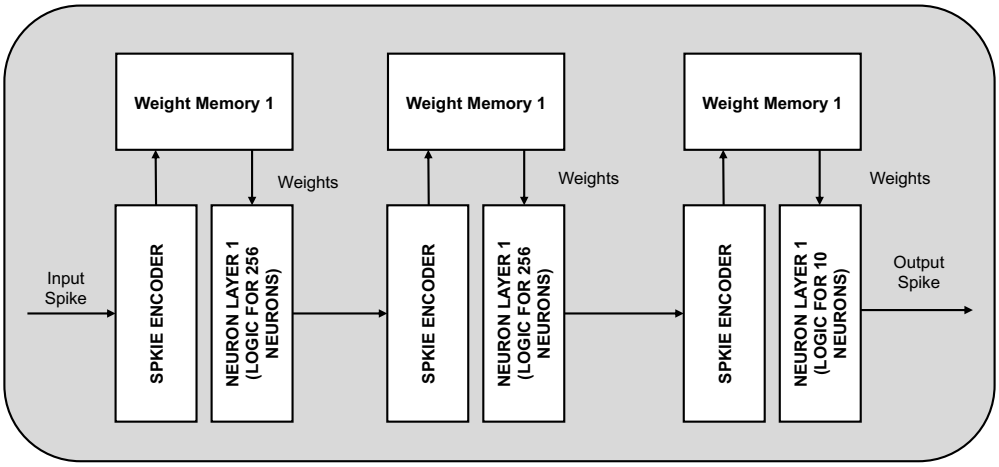

Figure 7. Block diagram of the neuromorphic hardware proposed by Yin et al. [49]. The architecture achieves a high level of energy efficiency by fixing the size of the network to a three-layer, 256-256-10 configurations. The input spikes for each layer are encoded by the spike encoder to select only the active neurons. Based on these indexes, corresponding weight values are fetched and loaded to the neurons calculation logic. Dedicated SRAM to store the weights of each layer is included.

Zheng et al. [50] proposed a low-power, event-driven neuromorphic hardware architecture which targets the evaluation of small-scale, fully connected SNNs trained online with weight-dependent STDP algorithms. The authors made two key contributions to reduce energy consumption. First, a novel modified STDP learning rule is presented to reduce the hardware complexity. Secondly, a software–hardware co-design approach, which leverages the sparsity of the spike trains and reuses the local memory storage for updating the local weights, helps to reduce the memory storage requirements and the latency of the learning algorithm. When implemented in a configuration of three-layer fully connected SNNs, the designed chip can reach a moderate accuracy on MNIST (around 90%) and energy consumption of 1.12 µJ per inference. The work opens up the possibilities of applying online training on embedded platforms; however, it suffers from low energy efficiency when compared to other works and the low accuracy on MNIST benchmarks.

Finally, the work by Chen et al. [51] is one of the most notable works in the field of low-power, embedded SNN accelerators. This work presents a reconfigurable neuromorphic hardware platform, implemented in a CMOS 10nm process, as described in Figure 8. The core contains 4096 highly reconfigurable neurons and could support up to one million synaptic connections. The core supports modeling for LIF neurons, on-chip STDP-based learning rules and off-chip learning rules. A high fan-out multicast network-on-chip is designed to support the communications between nodes. To reduce the energy cost, two key techniques are used. The first is to introduce structural sparsity in the learned weights, which could reduce the memory footprints for weights by 16x while introducing 2% overhead in control logic. The second is the approximate computing technique to drop some redundancies while computing with spike trains. On offline-trained MNIST configurations with 50% weight-sparsity, it could reach up to 97.7% accuracy at 1.7 µJ per classification.

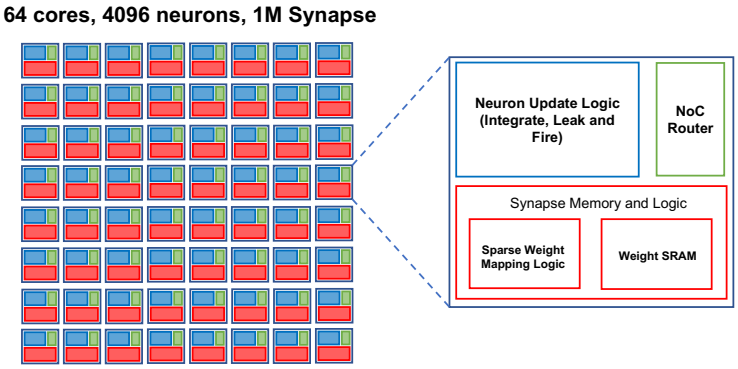

Figure 8. Block diagram of the neuromorphic chip proposed by Chen et al. [51]. The architecture consists of 64 neuromorphic cores, connected with a custom NoC. Each core has separate synapse memory and neuron update logic, which can support up to 64 neurons per core. High energy efficiency is achieved with a sparse connection mapping between neurons.

Table 2 describes a comparison between various low-power, embedded accelerators.

Table 2. Low-power SNN accelerators comparison.

Processor	Frenkel et al. [48]	Yin et al. [49]	Zheng et al. [50]	Chen et al. [51]
Implementation	Digital	Digital	Digital	Digital
Technology	28 nm	28 nm	65 nm	10 nm
Weight Resolution	4b	7b	16b	8b
Online learning	Yes	No	Yes	Yes
Networks models	FC 1 layer	FC 3 layers	FC 3 layers	FC 4 layers
Input coding scheme	Rate coding	Rate coding	Rate coding	Rate coding
MNIST accuracy	85.4%	98.7%	90%	97.9%
Core Area (mm^2)	16	1	4096	128
Energy/classification	15 nJ	773 nJ	1.12 µJ	1.7 µJ

5. Future Possibilities for Spiking Neural Networks

In this section, we highlight the possibilities to improve the performance of Spiking Neural Networks, both from a software point of view (improving training algorithms) and from new trends of hardware implementations.

Figure 9 shows the software point of view to improve the performance of Spiking Neural Networks. In fact, finding efficient training algorithms is one of the major challenges in the neuromorphic research field. The challenges come from two major problems. The first is that, currently, the scope of applications for SNNs is quite limited. Most of the current works in SNNs report the performance on the MNIST image recognition dataset, which is considered a classic problem and has been considered trivial in the formal DNN network. Even so, the best-reported results of SNNs are still behind those of the state-of-the-art DNNs. This has been credited to the lack of a specific dataset for the neuromorphic platform, as the MNIST dataset still needs to be explicitly converted to spike trains [52]. The second major problem with the current learning algorithms is that they normally require many simplifications of the network models and neuron models to be able to perform well, hence a particular training method is only suitable for a specific task. A consensus has yet to be reached among researchers on the universal approach to train SNNs.

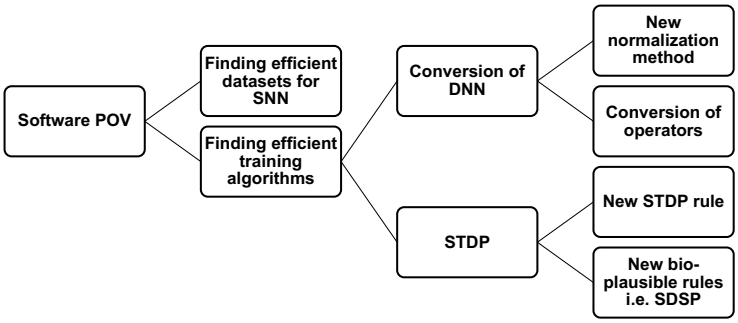

Figure 9. Future possibilities for SNN improvements, from a software point of view.

However, there are many opportunities to improve as well. For the direct conversion from ANN to SNN approach, the current trends of research are to find normalization techniques of weight and threshold to reduce the accuracy loss and to find novel ways to successfully convert all the techniques from DNNs models to SNN models. For the unsupervised learning approach, the major drawback is that the STDP-based learning algorithms do not permit the training of deep networks with many layers. One possible research direction is to apply some kind of supervised mechanism to the training such as reinforcement learning [53], preprocess the data [54] or modify the learning rule to allow communications between layers [55]. The most promising solution to the training algorithm, which has received much interest over the years, is the direct supervised learning method, based on the backpropagation technique. However, it is not suitable for many kinds of networks as it often requires modifications of neuron and network models to allow the backward flow of gradients.

For hardware improvements, several possible improvements could be made to enhance the energy efficiency of SNN operations as shown in Figure 10. As the current state-of-the-art SNNs require high bit widths to represent weights, weight quantization techniques to reduce the memory storage footprints while preserving accuracies are preferred. Another challenge is the high latency of SNN operations. Neuromorphic platforms could potentially offer better energy efficiency when compared to traditional DNN platforms. However, state-of-the-art SNNs often require many algorithmic timesteps to reach peak accuracies, increasing the number of computations needed. To better suit the embedded platforms, new SNN models need to be developed which could lead to fast classification in only a few timesteps. Another research trend which could be investigated is the new technologies for neuromorphic hardware platform, such as using superconducting nanowires for neurons and memristive crossbar array. In [56], Toomey et al. showed that, by using superconducting nanowires, it is possible to design a single neuron and a synapse system that exhibits a wide range of bio-realistic characteristics. It has been shown that the superconducting nanowires give four orders of magnitude improvements in terms of energy consumption when compared to other CMOS implementations while supporting a large number of fan-outs. This could become the next building block for the future neuromorphic platform. However, the device density and the operating temperature remain challenges for future research. Another exciting direction is to use a crossbar array with memristive devices to implement neuromorphic computing [57–59]. Crossbar array could be used for both DNN and SNN hardware architectures to realize the dot product of weights and inputs [58], where the synapse weights and the neuron state update logic are kept close. The current memristive devices such as Phase Change Memory (PCM) [60] or metal-oxide-resistive-devices [61] give low power consumption but are not stable yet, thus leading to a drop of accuracy when mapping a neural network [62]. Most of the studies are still in an early stage of research with results obtained from simulations.

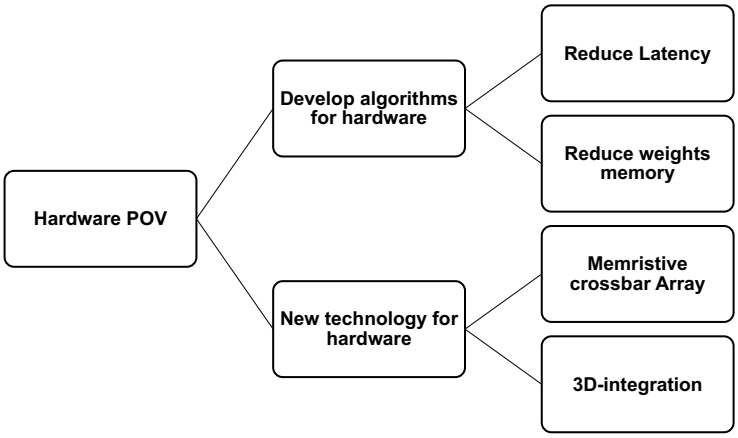

Figure 10. Future possibilities for SNN improvements, from a hardware point of view.

6. Conclusions

In this review, we present a brief introduction to the research field of SNNs, with both software (algorithms) and hardware (neuromorphic hardware implementations) perspectives. We briefly introduce the fundamentals of spiking neural networks, such as the neuron models, synapse models and some popular encoding strategies. We present and classify the major research directions in finding efficient training algorithms for SNNs. We conclude that unsupervised learning with STDP could offer a bio-plausible solution to applications that require constant changes to adapt to surrounding environments. However, the main limitation of this approach is that it only allows simple and shallow networks to be trained effectively. In contrast, the conversion from DNN to SNN could potentially leverage d a lot of pre-trained, complex networks in the literature, at the cost of limited flexibility in adapting to real-world applications. Recently, some direct training methods for SNN based on backpropagation have gained popularity, as they could reach the same accuracy as their DNN counterparts. However, to find a differentiable surrogate to the activation function of spiking neuron remains a challenge for future works.

We show that neuromorphic hardware implementations for SNNs can be broadly classified into two categories: (1) large-scale accelerators with a scalable architecture, which are suitable for very deep, brain-like network topologies; and (2) small-scale, low-power accelerators, which are suitable for embedded applications. With vastly different constraints in terms of power and energy consumption, the general hardware architectures and implementation strategies are also different. Finally, we discuss the possible improvements that could be made to algorithm research as well as the new trends in the development and implementation of truly neuromorphic hardware and devices.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Siem Reap, Cambodia, 13–16 December 2018.
2. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
3. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
4. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
5. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]

6. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
7. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
8. Gupta, A.; Long, L.N. Character Recognition using Spiking Neural Networks. In Proceedings of the 2007 International Joint Conference on Neural Networks, Orlando, FL, USA, 12–17 August 2007; pp. 53–58.
9. Meftah, B.; Lezoray, O.; Benyettou, A. Segmentation and Edge Detection Based on Spiking Neural Network Model. *Neural Process. Lett.* **2010**, *32*, 131–146. [CrossRef]
10. Escobar, M.J.; Masson, G.S.; Vieville, T.; Kornprobst, P. Action Recognition Using a Bio-Inspired Feedforward Spiking Network. *Int. J. Comput. Vis.* **2009**, *82*, 284. [CrossRef]
11. Tavanaei, A.; Maida, A. Bio-inspired Multi-layer Spiking Neural Network Extracts Discriminative Features from Speech Signals. In *Neural Information Processing*; Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S.M., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 899–908.
12. Loiselle, S.; Rouat, J.; Pressnitzer, D.; Thorpe, S. Exploration of rank order coding with spiking neural networks for speech recognition. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 4, pp. 2076–2080.
13. Ghosh-Dastidar, S.; Adeli, H. Improved Spiking Neural Networks for EEG Classification and Epilepsy and Seizure Detection. *Integr. Comput.-Aided Eng.* **2007**, *14*, 187–212. [CrossRef]
14. Kasabov, N.; Feigin, V.; Hou, Z.G.; Chen, Y.; Liang, L.; Krishnamurthi, R.; Othman, M.; Parmar, P. Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. *Neurocomputing* **2014**, *134*, 269–279. [CrossRef]
15. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1952**, *117*, 500–544. [CrossRef] [PubMed]
16. Izhikevich, E.M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **2003**, *14*, 1569–1572. [CrossRef]
17. Cassidy, A.S.; Merolla, P.; Arthur, J.V.; Esser, S.K.; Jackson, B.; Alvarez-Icaza, R.; Datta, P.; Sawada, J.; Wong, T.M.; Feldman, V.; et al. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–10. [CrossRef]
18. Indiveri, G.; Linares-Barranco, B.; Hamilton, T.; van Schaik, A.; Etienne-Cummings, R.; Delbruck, T.; Liu, S.C.; Dudek, P.; Häfliger, P.; Renaud, S.; et al. Neuromorphic Silicon Neuron Circuits. *Front. Neurosci.* **2011**, *5*, 73. [CrossRef]
19. Camunas-Mesa, L.; Acosta-Jimenez, A.; Serrano-Gotarredona, T.; Linares-Barranco, B. Fully digital AER convolution chip for vision processing. In Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS), Seattle, WA, USA, 18–21 May 2008; pp. 652–655. [CrossRef]
20. Nguyen, D.A.; Bui, D.H.; Iacopi, F.; Tran, X.T. An Efficient Event-driven Neuromorphic Architecture for Deep Spiking Neural Networks. In Proceedings of the 2019 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 3–6 September 2019; pp. 144–149. [CrossRef]
21. Haghiri, S.; Naderi, A.; Ghanbari, B.; Ahmadi, A. High Speed and Low Digital Resources Implementation of Hodgkin-Huxley Neuronal Model Using Base-2 Functions. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**. [CrossRef]
22. Andreev, V.; Ostrovskii, V.; Karimov, T.; Tutueva, A.; Doynikova, E.; Butusov, D. Synthesis and Analysis of the Fixed-Point Hodgkin–Huxley Neuron Model. *Electronics* **2020**, *9*, 434. [CrossRef]
23. Levi, T.; Khoyratee, F.; Saïghi, S.; Ikeuchi, Y. Digital implementation of Hodgkin–Huxley neuron model for neurological diseases studies. *Artif. Life Robot.* **2018**, *23*, 10–14. [CrossRef]
24. Yaghini Bonabi, S.; Asgharian, H.; Safari, S.; Nili Ahmadabadi, M. FPGA implementation of a biological neural network based on the Hodgkin-Huxley neuron model. *Front. Neurosci.* **2014**, *8*, 379. [CrossRef] [PubMed]
25. Pu, J.; Goh, W.L.; Nambiar, V.P.; Chong, Y.S.; Do, A.T. A Low-Cost High-Throughput Digital Design of Biorealistic Spiking Neuron. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**. [CrossRef]
26. Soleimani, H.; Ahmadi, A.; Bavandpour, M. Biologically inspired spiking neurons: Piecewise linear models and digital implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2012**, *59*, 2991–3004. [CrossRef]
27. Leigh, A.J.; Mirhassani, M.; Muscedere, R. An Efficient Spiking Neuron Hardware System Based on the Hardware-Oriented Modified Izhikevich Neuron (HOMIN) Model. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 3377–3381. [CrossRef]
28. Kumar, A.; Rotter, S.; Aertsen, A. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nat. Rev. Neurosci.* **2010**, *11*, 615–627. [CrossRef]
29. Rueckauer, B.; Liu, S.C. Conversion of analog to spiking neural networks using sparse temporal coding. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
30. Reich, D.S.; Mechler, F.; Purpura, K.P.; Victor, J.D. Interspike Intervals, Receptive Fields, and Information Encoding in Primary Visual Cortex. *J. Neurosci.* **2000**, *20*, 1964–1974. [CrossRef]
31. Caporale, N.; Dan, Y. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annu. Rev. Neurosci.* **2008**, *31*, 25–46. [CrossRef]

32. Markram, H.; Gerstner, W.; Sjöström, P.J. A history of spike-timing-dependent plasticity. *Front. Synaptic Neurosci.* **2011**, *3*, 4. [CrossRef]
33. Dan, Y.; Poo, M.M. Spike Timing-Dependent Plasticity: From Synapse to Perception. *Physiol. Rev.* **2006**, *86*, 1033–1048. [CrossRef] [PubMed]
34. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002.
35. Bohte, S.M.; Kok, J.N.; Poutré, H.L. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **2002**, *48*, 17–37. [CrossRef]
36. Lee, J.H.; Delbruck, T.; Pfeiffer, M. Training Deep Spiking Neural Networks Using Backpropagation. *Front. Neurosci.* **2016**, *10*, 508. [CrossRef] [PubMed]
37. Mostafa, H. Supervised Learning Based on Temporal Coding in Spiking Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3227–3235. [CrossRef] [PubMed]
38. Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Shi, L. Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Front. Neurosci.* **2018**, *12*, 331. [CrossRef] [PubMed]
39. Hu, Y.; Tang, H.; Wang, Y.; Pan, G. Spiking Deep Residual Network. *arXiv* **2018**, arXiv:1805.01352.
40. Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; Roy, K. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Front. Neurosci.* **2019**, *13*, 95. [CrossRef]
41. Pérez-Carrasco, J.A.; Zhao, B.; Serrano, C.; Acha, B.; Serrano-Gotarredona, T.; Chen, S.; Linares-Barranco, B. Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing–Application to Feedforward ConvNets. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2706–2719. [CrossRef]
42. Diehl, P.U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.; Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8. [CrossRef]
43. Cao, Y.; Chen, Y.; Khosla, D. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *Int. J. Comput. Vis.* **2015**, *113*, 54–66. [CrossRef]
44. Furber, S.B.; Galluppi, F.; Temple, S.; Plana, L.A. The SpiNNaker Project. *Proc. IEEE* **2014**, *102*, 652–665. [CrossRef]
45. Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.; et al. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1537–1557. [CrossRef]
46. Benjamin, B.V.; Gao, P.; McQuinn, E.; Choudhary, S.; Chandrasekaran, A.R.; Bussat, J.M.; Alvarez-Icaza, R.; Arthur, J.V.; Merolla, P.A.; Boahen, K. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* **2014**, *102*, 699–716. [CrossRef]
47. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **2018**, *38*, 82–99. [CrossRef]
48. Frenkel, C.; Lefebvre, M.; Legat, J.D.; Bol, D. A 0.086-mm^2 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS. *IEEE Trans. Biomed. Circuits Syst.* **2019**, *13*, 145–158. [CrossRef]
49. Yin, S.; Venkataramanaiah, S.K.; Chen, G.K.; Krishnamurthy, R.; Cao, Y.; Chakrabarti, C.; Seo, J. Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations. In Proceedings of the 2017 IEEE Biomedical Circuits and Systems Conference (BioCAS), Turin, Italy, 19–21 October 2017; pp. 1–5. [CrossRef]
50. Zheng, N.; Mazumder, P. A Low-Power Hardware Architecture for On-Line Supervised Learning in Multi-Layer Spiking Neural Networks. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5. [CrossRef]
51. Chen, G.K.; Kumar, R.; Sumbul, H.E.; Knag, P.C.; Krishnamurthy, R.K. A 4096-Neuron 1M-Synapse 3.8-pJ/SOP Spiking Neural Network With On-Chip STDP Learning and Sparse Weights in 10-nm FinFET CMOS. *IEEE J. Solid-State Circuits* **2019**, *54*, 992–1002. [CrossRef]
52. Deng, L.; Wu, Y.; Hu, X.; Liang, L.; Ding, Y.; Li, G.; Zhao, G.; Li, P.; Xie, Y. Rethinking the performance comparison between SNNS and ANNS. *Neural Netw.* **2020**, *121*, 294–307. [CrossRef]
53. Mozafari, M.; Ganjtabesh, M.; Nowzari-Dalini, A.; Thorpe, S.J.; Masquelier, T. Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition. *arXiv* **2018**, arXiv:1804.00227.
54. Kheradpisheh, S.R.; Ganjtabesh, M.; Masquelier, T. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing* **2016**, *205*, 382–392. [CrossRef]
55. Thiele, J.C.; Bichler, O.; Dupret, A. Event-based, timescale invariant unsupervised online deep learning with STDP. *Front. Comput. Neurosci.* **2018**, *12*, 46. [CrossRef]
56. Toomey, E.; Segall, K.; Berggren, K.K. Design of a Power Efficient Artificial Neuron Using Superconducting Nanowires. *Front. Neurosci.* **2019**, *13*, 933. [CrossRef] [PubMed]
57. Burr, G.; Narayanan, P.; Shelby, R.; Sidler, S.; Boybat, I.; di Nolfo, C.; Leblebici, Y. Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power). In Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM), Washington, DC, USA, 7–9 December 2015; p. 4.

58. Burr, G.W.; Shelby, R.M.; Sebastian, A.; Kim, S.; Kim, S.; Sidler, S.; Virwani, K.; Ishii, M.; Narayanan, P.; Fumarola, A.; et al. Neuromorphic computing using non-volatile memory. *Adv. Phys. X* **2017**, *2*, 89–124. [CrossRef]
59. Ankit, A.; Sengupta, A.; Panda, P.; Roy, K. Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks. In Proceedings of the 54th Annual Design Automation Conference 2017, Austin, TX, USA, 18–22 June 2017; pp. 1–6.
60. Kim, S.; Ishii, M.; Lewis, S.; Perri, T.; BrightSky, M.; Kim, W.; Jordan, R.; Burr, G.; Sosa, N.; Ray, A.; et al. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM), Washington, DC, USA, 7–9 December 2015.
61. Prezioso, M.; Merrikh-Bayat, F.; Hoskins, B.; Adam, G.C.; Likharev, K.K.; Strukov, D.B. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **2015**, *521*, 61–64. [CrossRef]
62. Kim, S.; Lim, M.; Kim, Y.; Kim, H.D.; Choi, S.J. Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network. *Sci. Rep.* **2018**, *8*, 1–7. [CrossRef] [PubMed]