Contents lists available at

Neural Networks

journal homepage:

RP-2

Review

Deep learning in spiking neural networks

Amirhossein Tavanaei [a], Masoud Ghodrati [b], Saeed Reza Kheradpisheh [c], Timothée Masquelier [d], Anthony Maida [a]

[a] School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70504, USA
[b] Department of Physiology, Monash University, Clayton, VIC, Australia
[c] Department of Computer Science, Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran
[d] CERCO UMR 5549, CNRS–Université de Toulouse 3, F-31300, France

ARTICLE INFO

Article history:
Received 7 June 2018
Received in revised form 2 December 2018
Accepted 3 December 2018
Available online 18 December 2018

Keywords:
Deep learning
Spiking neural network
Biological plausibility
Machine learning
Power-efficient architecture

ABSTRACT

In recent years, deep learning has revolutionized the field of machine learning, for computer vision in particular. In this approach, a deep (multilayer) artificial neural network (ANN) is trained, most often in a supervised manner using backpropagation. Vast amounts of labeled training examples are required, but the resulting classification accuracy is truly impressive, sometimes outperforming humans.

Neurons in an ANN are characterized by a single, static, continuous-valued activation. Yet biological neurons use discrete spikes to compute and transmit information, and the spike times, in addition to the spike rates, matter. Spiking neural networks (SNNs) are thus more biologically realistic than ANNs, and are arguably the only viable option if one wants to understand how the brain computes at the neuronal description level. The spikes of biological neurons are sparse in time and space, and event-driven. Combined with bio-plausible local learning rules, this makes it easier to build low-power, neuromorphic hardware for SNNs. However, training deep SNNs remains a challenge. Spiking neurons' transfer function is usually non-differentiable, which prevents using backpropagation.

Here we review recent supervised and unsupervised methods to train deep SNNs, and compare them in terms of accuracy and computational cost. The emerging picture is that SNNs still lag behind ANNs in terms of accuracy, but the gap is decreasing, and can even vanish on some tasks, while SNNs typically require many fewer operations and are the better candidates to process spatio-temporal data.

Contents

1. Introduction

Artificial neural networks (ANNs) are predominantly built using idealized computing units with continuous activation values and a

* Corresponding author.
E-mail address: (A. Tavanaei).

prosthetic limb →

instead of using SNN with only one layer (eg) to implement SNN with multiple layer → [improve] performance potential.

set of weighted in
because of their
rons use different
linear activation
to stack more tha
makes it possibl
for training. Wit
data sets, compu
computing, and
have become st
generalize unse
performance of

A distinct br
in the ILSVRC image clas-
sification challenge
AlexNet became
known as a deep neural network (DNN) because it consisted of
about eight sequential layers of end-to-end learning, totaling 60
million trainable parameters. For recent reviews of DNNs, see
and
). DNNs
have been remarkably successful in many applications including
image recognition
), object
detection
), speech recognition (
),
biomedicine and bioinformatics (
), temporal data process-
ing (
), and many other applications (
).
These recent advances in artificial intelligence (AI) have opened up
new avenues for developing different engineering applications and
understanding of how biological brains work (
;
).

Although DNNs are historically brain-inspired, there are fundamental differences in their structure, neural computations, and learning rule compared to the brain. One of the most important differences is the way that information propagates between their units. It is this observation that leads to the realm of spiking neural networks (SNNs). In the brain, the communication between neurons is done by broadcasting trains of action potentials, also known as spike trains to downstream neurons. These individual spikes are sparse in time, so each spike has high information content, and to a first approximation has uniform amplitude (100 mV with spike width about 1 msec). Thus, information in SNNs is conveyed by spike timing, including latencies, and spike rates, possibly over populations (
). SNNs almost universally use idealized spike generation mechanisms in contrast to the actual biophysical mechanisms (
).

ANNs, that are non-spiking DNNs, communicate using continuous valued activations. Although the energy efficiency of DNNs can likely be improved, SNNs offer a special opportunity in this regard because, as explained below, spike events are sparse in time. Spiking networks also have the advantage of being intrinsically sensitive to the temporal characteristics of information transmission that occurs in the biological neural systems. It has been shown that the precise timing of every spike is highly reliable for several areas of the brain and suggesting an important role in neural coding (
;
;
). This precise temporal pattern in spiking activity is considered as a crucial coding strategy in sensory information processing areas (
;
;
;
;
;
) and neural motor control areas in the brain (
;
;
). SNNs have become the focus of a number of recent applications in many areas

rtificial Networks 111 (2019) 47–63

ttern recognition such as visual processing (
), speech
gnition (
), and medical
nosis (
). In
nt years, a new generation of neural networks that incorpo-
s the multilayer structure of DNNs (and the brain) and the type
formation communication in SNNs has emerged. These deep
s are great candidates to investigate neural computation and
rent coding strategies in the brain.

In regard to the scientific motivation, it is well accepted that the ability of the brain to recognize complex visual patterns or identify auditory targets in a noisy environment is a result of several processing stages and multiple learning mechanisms embedded in deep spiking networks (
). In comparison to traditional deep networks, training deep spiking networks is in its early phases. It is an important scientific question to understand how such networks can be trained to perform different tasks as this can help us to generate and investigate novel hypotheses, such as rate versus temporal coding, and develop experimental ideas prior to performing physiological experiments. On the other hand, SNNs provide neural architectures to better process spatio-temporal data, especially in an online mode (
) and temporal binary activation of digital systems (
). Knowledge representation in time and space makes SNNs unique to perform brain-like computations and to understand the brain data/activity in a spatio-temporal pattern. More details about the knowledge representation provided by SNNs have been discussed in
).

In regard to the engineering motivation, SNNs have some advantages over traditional neural networks in regard to implementation in special purpose hardware. At the present, effective training of traditional deep networks requires the use of energy intensive high-end graphic cards. Spiking networks have the interesting property that the output spike trains can be made sparse in time. An advantage of this in biological networks is that the spike events consume energy and that using few spikes which have high information content reduces energy consumption (
). This same advantage is maintained in hardware (
;
;
). Thus, it is possible to create low energy spiking hardware which is highly responsive to event-based sensors based on the property that spikes are sparse in time (
).

An important part of the learning in deep neural models, both spiking and non-spiking, occurs in the feature discovery hierarchy, where increasingly complex, discriminative, abstract, and invariant features are acquired (
). Given the scientific and engineering motivations mentioned above, deep SNNs provide appropriate architectures for developing an efficient, brain-like representation. Also, pattern recognition in the primate's brain is done through multi-layer neural circuits that communicate by spiking events. This naturally leads to interest in using artificial SNNs in applications that brains are good at, such as pattern recognition (
). Bio-inspired SNNs, in principle, have higher representation power and capacity than traditional rate-coded networks (
). Furthermore, SNNs allow a type of bio-inspired learning (weight modification) that depends on the relative timing of spikes between pairs of directly connected neurons in which the information required for weight modification is locally available. This local learning resembles the remarkable learning that occurs in many areas of the brain (
;
;
;
;
).

- Bio-inspired neurons (weight modification)

The spike trains are represented formally by sums of Dirac delta functions and do not have derivatives. This makes it difficult to use derivative-based optimization for training SNNs, although very recent work has explored the use of various types of substitute or approximate derivatives (;). This raises a question: How are neural networks in the brain trained if derivative-based optimization is not available? Although spiking networks have theoretically been shown to have Turing-equivalent computing power ( ), it remains a challenge to train SNNs, especially deep SNNs using multi-layer learning. In many existing spiking networks, learning is restricted to a single layer, for example (; ; ; ;). Equipping spiking networks with multi-layer learning is an open area that has potential to greatly improve their performance on different tasks. The main core of the previous research is based on the fact that coding with the timing of spikes carries useful information and exhibits great computational power in biological systems (; ; ;).

Here, we review recent studies in developing deep learning models in SNNs with the focus on: (1) describing the SNNs' architectures and their learning approaches; (2) reviewing deep SNNs composed of feedforward, fully connected spiking neural layers; (3) spiking convolutional neural networks; (4) reviewing spiking restricted Boltzmann machines and spiking deep belief networks; (5) reviewing recurrent SNNs; and (6) providing a comprehensive summary comparing the performance of recent deep spiking networks. We hope that this review will help researchers in the area of artificial neural networks to develop and extend efficient and high-performance deep SNNs and will also foster a cross-fertilization in future experimental and theoretical work in neuroscience.

2. Spiking Neural Networks: A biologically inspired approach to information processing

The introduction of SNNs in the last few decades, as a powerful third generation neural network (,), has encouraged many studies with the focus on biologically motivated approaches for pattern recognition (; ;). SNNs were originally inspired by the brain and the communication scheme that neurons use for information transformation via discrete action potentials (spikes) in time through adaptive synapses. In a biological neuron, a spike is generated when the running sum of changes in the membrane potential, which can result from presynaptic stimulation, crosses a threshold. The rate of spike generation and the temporal pattern of spike trains carry information about external stimuli (; ;) and ongoing calculations. SNNs use a very similar process for spike generation and information transformation. In the following sections, we explain the details of SNN architectures and learning methods applied to these types of networks.

2.1. SNN architecture

An SNN architecture consists of spiking neurons and interconnecting synapses that are modeled by adjustable scalar weights. The first step in implementing an SNN is to encode the analog input data into the spike trains using either a rate based method (; ;), some form of temporal coding (; ;), or population coding (; ;). As stated earlier, a biological neuron in the brain (and similarly in a simulated spiking neuron) receives synaptic inputs form other neurons in the neural network. Biological neural networks have both action potential generation dynamics and network dynamics. In comparison to true biological networks, the network dynamics of artificial SNNs are highly simplified. In this context, it is useful to assume that the modeled spiking neurons have pure threshold dynamics (in contrast to, e.g., refractoriness, hysteresis, resonance dynamics, or post-inhibitory rebound properties). The activity of pre-synaptic neurons modulates the membrane potential of postsynaptic neurons, generating an action potential or spike when the membrane potential crosses a threshold. Hodgkin and Huxley were the first to model this phenomenon (,). Specifically, they created a model of action potential generation from the voltage gating properties of the ion channels in the squid cell membrane of the squid axon. After the Hodgkin and Huxley model with extensive biological details and high computational cost (; ;), diverse neuron models have been proposed such as the spike response model (SRM) (;), the Izhikevich neuron model (,), and the leaky integrated-and-fire (LIF) neuron (;). The LIF model is extremely popular because it captures the intuitive properties of external input accumulating charge across a leaky cell membrane with a clear threshold.

Spike trains in a network of spiking neurons are propagated through synaptic connections. A synapse can be either excitatory, which increases the neuron's membrane potential upon receiving input, or inhibitory, which decreases the neuron's membrane potential (;). The strength of the adaptive synapses (weights) can be changed as a result of learning. The learning rule of an SNN is its most challenging component for developing multi-layer (deep) SNNs, because the non-differentiability of spike trains limits the popular backpropagation algorithm.

2.2. Learning rules in SNNs

As previously mentioned, in virtually all ANNs, spiking or non-spiking, learning is realized by adjusting scalar-valued synaptic weights. Spiking enables a type of bio-plausible learning rule that cannot be directly replicated in non-spiking networks. Neuroscientists have identified many variants of this learning rule that falls under the umbrella term spike-timing-dependent plasticity (STDP). Its key feature is that the weight (synaptic efficacy) connecting a pre- and post-synaptic neuron is adjusted according to their relative spike times within an interval of roughly tens of milliseconds in length (,). The information used to perform the weight adjustment is both local to the synapse and local in time. The following subsections describe common learning mechanisms in SNNs, both unsupervised and supervised.

2.2.1. Unsupervised learning via STDP

As stated above, unsupervised learning in SNNs often involves STDP as part of the learning mechanism (; ;). The most common form of biological STDP has a very intuitive interpretation. If a presynaptic neuron fires briefly (e.g., ≈ 10 ms) before the postsynaptic neuron, the weight connecting them is strengthened. If the presynaptic neuron fires briefly after the postsynaptic neuron, then the causal relationship between the temporal events is spurious and the weight is weakened. Strengthening is called long-term potentiation (LTP) and weakening is called long-term depression (LTD). The phrase "long-term" is used to distinguish between very transient effects on the scale of a few ms that are observed in experiments.

Formula idealizes the most common experimentally observed STDP rule for a single pair of spikes obtained by fitting to experimental data (,).

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} \leq 0 \quad A > 0 \\ Be^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} > 0 \quad B < 0 \end{cases} \tag{1}$$

w is the synaptic weight. $A_+ > 0$ and $A_- > 0$ are usually constant parameters indicating learning rates. τ is the time constant (e.g. 15 ms) for the temporal learning window. The first of the above cases describes LTP while the second describes LTD. The strength of the effect is modulated by a decaying exponential whose magnitude is controlled by the time-constant-scaled time difference between the pre- and postsynaptic spikes. Rarely, do artificial SNNs use this exact rule. They usually use a variant, either to achieve more simplicity or to satisfy a convenient mathematical property.

Besides the temporally and spatially local weight change described in Eq. , STDP has known important temporally accumulated network-level effects. For instance, STDP affects a neuron's behavior in response to repeated spike patterns embedded in a possibly stochastic spike train. A neuron (equipped with STDP-trained synapses) in coincidence with similar volleys of spikes is able to concentrate on afferents that consistently fire early (shorter latencies) (;). Spike trains in many areas of the brain are highly reproducible. ( ) have shown that presenting repeated inputs to an SNN equipped with STDP shapes neuronal selectivity to the stimulus patterns within the SNN. Specifically, they showed that the response latency of the postsynaptic potential is decreased as STDP proceeds. Reducing the postsynaptic latency results in faster neural processing. Thus, the neuron responds faster to a specific input pattern than to any other. In fact, the STDP rule focuses on the first spikes of the input pattern which contain most of the information needed for pattern recognition. It has been shown that repeating spatio-temporal patterns can be detected and learned by a single neuron based on STDP (; ;). STDP can also solve difficult computational problems in localizing a repeating spatio-temporal spike pattern and enabling some forms of temporal coding, even if an explicit time reference is missing (;). Using this approach, more complex networks with multiple output neurons have been developed (; ; ; ;).

2.2.2. Probabilistic characterization of unsupervised STDP

Many studies offer evidence that at least an approximate Bayesian analysis of sensory stimuli occurs in the brain (; ; ; ; ;). In Bayesian inference, hidden causes (such as presence of an object of a particular category) are inferred using both prior knowledge and the likelihood of new observations to obtain a posterior probability of the possible cause. Researchers have considered the possible role of probabilistic (Bayesian) computation as a primary information processing step in the brain in terms of STDP.

( ) showed that a form of STDP, when used with Poisson spiking input neurons coupled with the appropriate stochastic winner-take-all (WTA) circuit, is able to approximate a stochastic online expectation maximization (EM) algorithm to learn the parameters for a multinomial mixture distribution. The model was intended to have some biological plausibility. The STDP rule used in their network is shown in Eq. . LTP occurs if the presynaptic neuron fires briefly (e.g., within ϵ = 10 ms) before the postsynaptic neuron. Otherwise LTD occurs. Generating a spike by an output neuron creates a sample from the coded posterior distribution of hidden variables which can be considered as the E-step in the EM algorithm. The application of STDP to the synapses of fired output neurons specifies the M-step in EM. ( ) extended their network by using an inhibitory neuron to implement the WTA in order to improve the compatibility of the model for embedding in a cortical microcircuit.

$$\Delta w_{ji} = \begin{cases} e^{-w_{ji}}, & 0 < t_i^f - t_j^f < \epsilon \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

Building on the stochastic WTA circuits described above ( ) developed a liquid state machine (LSM, Section) containing input neurons, a reservoir of the WTA circuits, and a linear output readout. Further extension showed that STDP, applied on both the lateral excitatory synapses and synapses from afferent neurons, can represent the underlying statistical structure of such spatio-temporal input patterns (;). In this framework, each spike train generated by the WTA circuits can be viewed as a sample from the state space of a hidden Markov model (HMM).

One drawback of the STDP model introduced in (;) is that its excitatory synaptic weights are negative. This, however, can be solved by shifting the weights to a positive value by using a constant parameter in the LTP rule. Based on this idea, (;) proposed an unsupervised rule for spatio-temporal pattern recognition and spoken word classification. It has been shown that the EM acquired in an SNN is able to approximately implement the EM algorithm in a Gaussian mixture model (GMM) embedded in the HMM states (;).

Using probabilistic learning in spiking neurons for modeling hidden causes has recently attracted attention. ( ) developed a bio-plausible learning rule based on the joint distribution of perceptions and hidden causes to adapt spontaneous spike sequences to match the empirical distribution of actual spike sequences (;). The learning strategy involved minimizing the Kullback–Leibler divergence (;) as a non-commutative distance measure between the distribution representing the model (SNN) and a target distribution (observation). The EM algorithm in recurrent SNNs (;) and probabilistic association between neurons generated by STDP in combination with intrinsic plasticity (;) are two other instances of probabilistic learning in SNNs. The probabilistic rules also have been employed in sequential data processing (;) and Markov chain Monte Carlo sampling interpreted by stochastic firing activity of spiking neurons (;).

2.2.3. Supervised learning

All supervised learning uses labels of some kind. Most commonly, supervised learning adjusts weights via gradient descent on a cost function comparing observed and desired network outputs. In the context of SNNs, supervised learning tries to minimize the error between desired and output spike trains, sometimes called readout error, in response to inputs.

SNN issues in relation to backpropagation. From a biological vantage point, there has been considerable skepticism about whether the backpropagation training procedure can be directly implemented in the brain. With respect to SNNs, there are two prominent issues which can be seen from the formula below. Shown below is a core formula, obtained from the chain rule, that occurs in all variants of backpropagation (;).

$$\delta_j^\mu = g'(a_j^\mu) \sum_k w_{kj} \delta_k^\mu \tag{3}$$

In the above, δ_j^μ and δ_k^μ denote the partial derivative of the cost function for input pattern μ with respect to the net input to some arbitrary unit j or k. Unit j projects direct feedforward connections to the set of units indexed by k. $g(\cdot)$ is the activation function

applied to the net input of unit j, where that net input is denoted a_j^n. w_{kj} are the feedforward weights projecting from unit j to the set of units indexed by k.

Both parts of the RHS of Eq. present complications for bio-plausible spiking versions of backpropagation. First, the expression $g_j(\cdot)$ requires $g_j(\cdot)$ with respect to w_{kj}. Since $g(\cdot)$ applies to a spiking neuron, it is likely represented by a sum of Dirac delta functions, which means the derivative does not exist. The second, and more serious complication, applies to both spiking and non-spiking networks and was apparently first pointed out by (, p. 49) and termed the "weight transport" problem. The problem is the following. The expression $\sum_k w_{kj}\delta_k^n$ uses the feedforward weights w_{kj} in a feedback fashion. This means that matching symmetric feedback weights must exist and project accurately to the correct neurons (point-to-point feedback) in order for Eq. to be usable.

In the literature, the first issue has generally been addressed by using substitute or approximate derivatives. One must be aware that some of these solutions are not bio-plausible. For example, using the membrane potential of the presynaptic neuron as a surrogate becomes problematic because its value is not local to the synapse (cf. , Section of this review). These approaches, however, are still useful from both engineering and scientific standpoints.

Progress on the second issue has recently been made by ( ) and ( ). It was shown in ( ) that for some tasks, backpropagation could still perform well if random feedback weights were used. The authors in ( ) explored this further, examining three kinds of feedback (uniform, random, and symmetric). They found that simpler problems could be solved by any kind of feedback whereas complex problems needed symmetric feedback.

Another study that appears to make progress on the second issue is the introduction of mirrored STDP learning by ( ). This research built an STDP trained auto-encoder with matched-weight, point-to-point connections across layers. This could be an approach to implementing the infrastructure to support brain-plausible backpropagation. However, other complicating issues arise. One is stability of a large-scale brain architecture having strong feedback connections (no strong loops hypothesis (,)).

Some supervised learning methods for SNNs. SpikeProp (,) appears to be the first algorithm to train SNNs by backpropagating errors. Their cost function took into account spike timing and SpikeProp was able to classify non-linearly separable data for a temporally encoded XOR problem using a 3-layer architecture. One of their key design choices was to use Gerstner's (,) spike-response model (SRM) for the spiking neurons. Using the SRM model, the issue of taking derivatives on the output spikes of the hidden units was avoided because those units' responses could be directly modeled as continuous-valued PSPs applying to the output synapses that they projected to. One limitation of this work is that each output unit was constrained to discharge exactly one spike. Also, continuous variable values, such as in the temporally extended XOR problem, had to be encoded as spike-time delays which could be quite long.

Later advanced versions of SpikeProp, Multi-SpikeProp, were applicable in multiple spike coding (, ; ,). Using the same neural architecture of SpikeProp, new formulations of temporal spike coding and spike time errors have recently improved the spiking backpropagation algorithm (, ; ,). A recent implementation of backpropagation in SNNs was proposed by ( ) who developed spatio-temporal gradient descent in multi-layer SNNs.

More recent approaches to supervised training of SNNs include ReSuMe (remote supervised learning) (, ;), Chronotron (,), and SPAN (spike pattern association neuron) (,), among others. All of the above models consist of a single spiking neuron receiving inputs from many spiking presynaptic neurons. The goal is to train the synapses to cause the post-synaptic neuron to generate a spike train with desired spike times.

ReSuMe adapts the Widrow–Hoff (Delta) rule, originally used for non-spiking linear units, to SNNs. The Widrow–Hoff rule weight changes are proportional to the desired output minus the observed output, as shown below.

$$\Delta w = (y^d - y^o)x = y^d x - y^o x \tag{4}$$

where x is the presynaptic input and y^d and y^o are the desired and observed outputs, respectively. When expanded as shown on the RHS and reformulated for SNNs, the rule can be expressed as a sum of STDP and anti-STDP. That is, the rule for training excitatory synapses takes the form

$$\Delta w = \Delta w^{\mathrm{STDP}}(S^{\mathrm{in}}, S^d) + \Delta w^{\mathrm{aSTDP}}(S^{\mathrm{in}}, S^o). \tag{5}$$

In the above, Δw^{STDP} is a function of the correlation of the presynaptic and desired spike trains, whereas $\Delta w^{\mathrm{aSTDP}}$ depends on the presynaptic and observed spike trains. Because the learning rule uses the correlation between the teacher neuron (desired output) and the input neuron, there is not a direct physical connection. This is why the word "remote" is used in the phrase "remote supervised learning." Although it is not apparent in the above equation, the learning is constrained to fall with typical STDP eligibility windows.

The Chronotron was developed to improve on the Tempotron (,) which had the ability to train single neurons to recognize encodings by the precise timing of incoming spikes. The limitation of the Tempotron was that was restricted to outputting 0 or 1 spikes during a predetermined interval. Because of this, the output did not encode spike timing information. This precluded the ability of a Tempotron to meaningfully send its output to another Tempotron. The motivation of the Chronotron was similar to that of SpikeProp and its successors. The innovation of the Chronotron was to base the supervised training on a more sophisticated distance measure, namely the Victor–Purpura (VP) distance metric (,) between two spike trains. This metric is "the minimum cost of transforming one spike train into the other by creating, removing, or moving spikes". (, , p. 3) They adapted the VP distance so that it would be piecewise differentiable and admissible as a cost function to perform gradient descent with respect to the weights.

Similar to ReSuMe, the SPAN model develops its learning algorithm from the Widrow–Hoff rule. However, instead of adapting the rule to an SNN, SPAN makes the SNN compatible with Widrow–Hoff by digital-to-analog conversion of spike trains using alpha kernels of the form $te^{\frac{-t}{\tau}}$. As this is a common formula for modeling a postsynaptic potential, this step in effect converts all spikes to a linear summation of PSPs. Note that this is similar to the technique used in SpikeProp described at the beginning of this subsection. The learning rule can then be written as

$$\Delta w \propto \int \tilde{x}_i(\tilde{y}_d(t) - \tilde{y}_o(t))dt \tag{6}$$

where the tilde symbol indicates the analog version of the spike train and the bounds of integration cover the relevant local time interval.

In ( ), it was observed that the previous gradient-based learning methods all still had the constraint that the number and times of output spikes must be prespecified which placed limits on their applicability. They replaced the hard spike threshold with a narrow support gate function, $g(\cdot)$, such that $g(v) \geq 0$ and $\int g dv = 1$, and v is the membrane potential. Intuitively, this allows modeled postsynaptic currents to be released when the membrane potential approaches threshold leading to continuity in the spike generation mechanism. Experimentally, it was found that weight updates occurred near spike times "bearing close resemblance to reward-modulated STDP" (, p. 8), which enhances the biological relevance of the model.

In ( ), a supervised learning method was proposed (BP-STDP) where the backpropagation update rules were converted to temporally local STDP rules for multi-layer SNNs. This model achieved accuracies comparable to equal-sized conventional and spiking networks for the MNIST benchmark (see Section).

Another implementation of supervised learning in SNNs is based on optimizing the likelihood and probability of the postsynaptic spikes to match the desired ones. ( ) developed a model to optimize the likelihood of postsynaptic firing at one or several desired times. They proposed a modified version of the SRM neuronal model such that it uses a stochastic threshold on the membrane potential.

In another approach to supervised learning, each output neuron represents a class of data (pattern). Output neurons are in competition to be selected and responsive to the input patterns. In this approach, firing the target neuron causes STDP over the incoming synapses and firing the non-target neurons causes anti-STDP. This approach has successfully been used in SNNs for numerical data classification ( ), handwritten digit recognition (,), spoken digit classification (,), and reinforcement learning in SNNs (,). The sharp synaptic weight adaptation based on immediate STDP and anti-STDP results in fast learning.

3. Deep learning in SNNs

Deep learning uses an architecture with many layers of trainable parameters and has demonstrated outstanding performance in machine learning and AI applications (, ; ,). Deep neural networks (DNNs) are trained end-to-end by using optimization algorithms usually based on backpropagation. The multi-layer neural architecture in the primate's brain has inspired researchers to concentrate on the depth of non-linear neural layers instead of using shallow networks with many neurons. Also, theoretical and experimental results show better performance of deep rather than wide structures (, ; , ,). Deep neural networks extract complex features through sequential layers of neurons equipped with non-linear, differentiable activation functions to provide an appropriate platform for the backpropagation algorithm. depicts a deep NN architecture with several hidden layers.

For most classification problems, the output layer of a deep network uses a softmax module. The training vectors use a one-hot encoding. In a one-hot encoding each vector component corresponds to one of the possible classes. This vector is binary with exactly one component set to 1 that corresponds to the desired target class. The softmax module for the output layer guarantees that the values of each of the output units falls within the range (0, 1) and also sum to 1. This gives a set of mutually exclusive and exhaustive probability values. The softmax formula, sometimes called the normalized exponential, is given below

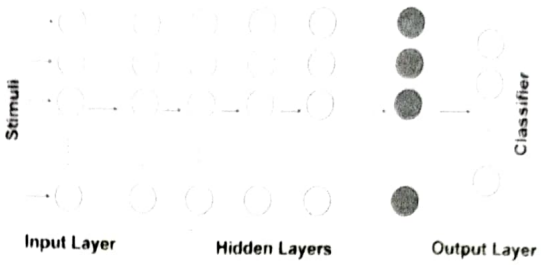

Fig. 1. Simplest deep neural architecture, usually fully connected, with input, hidden, and output layers. The input layer learns to perform pre-processing on the input. The information is then sent to a series of hidden layers, the number of which can vary. As the information propagates through hidden layers, more complex features are extracted and learned. The output layer performs classification and determines the label of the input stimulus, usually by softmax (see text).

$$y_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)} \tag{7}$$

where, a_i, is the net input to a particular output unit, j indexes the set of output units, and y_i is the value of output unit i, which falls in the range (0, 1).

In addition to the fully connected architecture in and discussed in Section , there are also deep convolutional neural networks (DCNNs) discussed in Section , deep belief networks (DBNs) discussed in Section , and recurrent neural networks (RNNs) discussed in Section .

SNNs have also shown promising performance in a number of pattern recognition tasks (, ; ,). However, the performance of directly trained spiking deep networks is not as good as traditional DNNs represented in the literature. Therefore, a spiking deep network (spiking DNN, spiking CNN, spiking RNN, or spiking DBN) with good performance comparable with traditional deep learning methods, is a challenging topic because of its importance in DNN hardware implementations. (,) developed one of the earliest feedforward hierarchical convolutional networks of spiking neurons for unsupervised learning of visual features (, ,). This network was extended for larger problems, such as ( ). Using an STDP rule with a probabilistic interpretation, the performance of the model was later improved in different object recognition tasks (,). Further attempts led to several multi-layer SNNs, with STDP learning, that performed greatly in adaptive multi-view pattern recognition (,) and handwritten digit recognition (,). These models mostly used one or more layers for pre-processing, one learning layer, and one classifier (output neuron) layer. Although these networks are known as multi-layer SNNs, they do not offer multi-layer learning. Specifically, these SNNs are limited by using only one trainable layer, even though they have many layers of processing.

Encouraged by the power-efficiency and biological plausibility of neuromorphic platforms, a number of recent studies have concentrated on developing deep SNNs for these platforms (,). Previous studies exploring supervised and unsupervised learning rules in spiking architectures can be employed to develop hierarchies of feature extraction and classification modules. SNNs enable power-efficient platforms mimicking brain functionality for solving complex problems, such as in new trends of autonomous vehicles. Developing a neural

network that is as efficient and biologically plausible as SNNs but as powerful as DNNs in performing different tasks is an opportunity in the field of artificial intelligence and computational neuroscience.

The remaining subsections review spiking deep learning approaches covering deep fully connected SNNs, spiking CNNs, spiking DBNs, and spiking RNNs.

3.1. Deep fully connected SNNs

Recent studies have developed a number of deep SNNs using STDP and stochastic gradient descent. Spiking networks consisting of many LIF neurons equipped by spike-based synaptic plasticity rules have shown success in different pattern recognition tasks (, , ,).

( ) showed that STDP in a two-layer SNN is able to extract discriminative features and patterns from stimuli. They used unsupervised learning rules introduced by ( ) and ( ) to train the SNN for the Modified National Institute of Standards and Technology (MNIST) dataset (, ' digit recognition with the best performance of 95%.

Towards linking biologically plausible learning methods and conventional learning algorithms in neural networks, a number of deep SNNs have recently been developed.

( ) proposed a deep learning method using forward and backward neural activity propagation. The learning rule was based on the idea that STDP implements the gradient descent learning rule (, ,). Using pre- and postsynaptic spike trains, ( ) developed a backpropagation algorithm in deep SNNs using the outer product of pre- and postsynaptic spike counts. They showed high performance of the spiking multi-layer perceptron on the MNIST benchmark (97.93%) which is comparable to the performance of the conventional deep neural networks equipped with rectified linear units (ReLUs) of 98.37%. Recently, ( ) proposed a backpropagation algorithm by treating the neuron's membrane potential as the differentiable signal to act analogous to the non-linear activation functions in traditional neural networks ( ). Performance of 98.88% on the MNIST dataset was reported in this study while the number of computational operations was five times fewer than traditional DNNs, in their experiments. To further reduce the computational cost of learning in these deep SNNs, ( ) proposed an event-driven random backpropagation (eRBP) algorithm simplifying the backpropagation chain path. The eRBP rule used error-modulated synaptic plasticity in which all the information used for learning was locally available at the neuron and synapse (,).

A more hardware-oriented approach to leveraging power-efficient SNNs is to convert an offline trained DNN to a neuromorphic spiking platform (ANN-to-SNN conversion), specifically for hardware implementation (,). To substitute for the floating-point activation values in DNNs, rate-based coding is generally used in which higher activations are replaced by higher spike rates. Using this approach, several models have been developed that obtained excellent accuracy performance (, : , ; , ; ,). In another effort to assess the power consumption of deep SNNs, ( ) studied many different models, all of which achieved the same accuracy rate of 98% on the MNIST digit recognition task. They all used the same 784-1200-1200-10 three-layer architecture but applied optimized parameters and SNN architecture settings, in ANN-to-SNN conversion, to reduce power consumption and latency of the model. The performance of a DNN and different converted deep SNN versus the total required operations is shown in . For the SNNs, a computational operation is "the addition of a synaptic weight to its neuron's membrane potential" (,). Some of the SNN variations explore a computation versus accuracy trade-off where fewer spikes incur less power consumption while preserving acceptable accuracy.

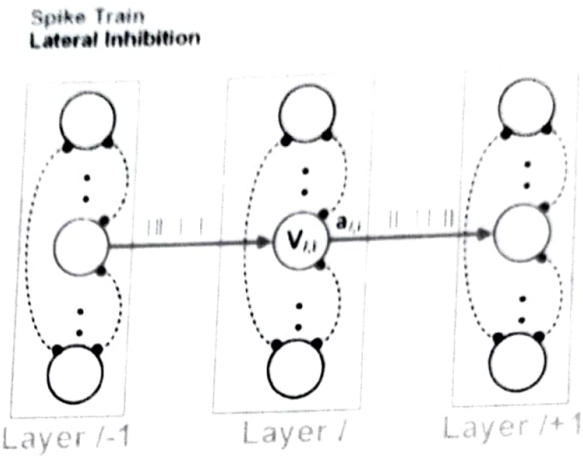

Fig. 2. Deep SNN equipped with backpropagation proposed by ( ). The neuron's activation value, $a_{i,l}$, is given by the neuron's membrane potential. The differentiable activation function, which is calculated by the neuron's excitatory input, lateral inhibition, and threshold, is used for developing backpropagation using the chain rule. The output activation value of the current layer (layer l) is used as input for the next layer in the backpropagation algorithm.

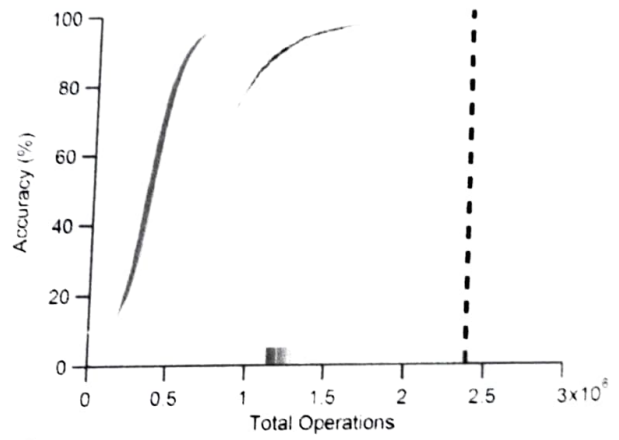

Fig. 3. The total number of operations needed to achieve a given accuracy for MNIST classification by deep SNNs converted from an offline trained deep neural network in comparison with the traditional (non-spiking) deep neural network (,). The vertical dashed line shows the number of operations required for the non-spiking deep neural network to achieve accuracy of 98%. The other curves show the accuracy of 522 deep SNNs (with different network setups) versus the number of operations. The pink curves show the networks that achieve less than 98% accuracy within the computing constraint. The colored vertical lines on the horizontal axis indicate the number of operations at which the corresponding SNNs reached 98% accuracy. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Spiking CNNs

Deep convolutional neural networks (DCNNs) are mostly used in applications involving images. They consist of a sequence of

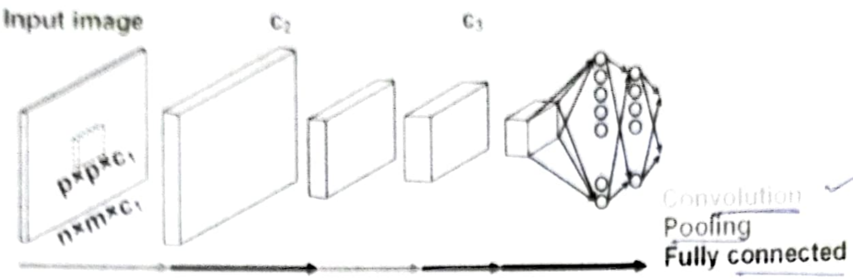

Fig. 4. LeNet: Early CNN proposed by ( ). The network consists of two convolutional/pooling layers followed by fully connected layers for image classification.

convolution and pooling (sub-sampling) layers followed by a feed-forward classifier like that in . This type of network has shown outstanding performance in image recognition ( ), speech recognition ( ), bioinformatics ( ), object detection and segmentation ( ), and so forth. shows the LeNet architecture, an early deep CNN, for image classification ( ).

The question is how a spiking CNN with such an architecture can be trained while incorporating traditional CNN properties. In the case of vision, the first layer of convolution is interpreted as extracting primary visual features (sometimes resembling oriented-edge detectors modeled by the outputs of Gabor filters ( )). Subsequent layers extract increasingly more complex features for classification purposes. The pooling layer performs subsampling and reduces the size of the previous layer using an arithmetic operation such as maximum or average over a square neighborhood of neurons in the relevant feature map. Later in the hierarchy, these layers develop invariance to changes in orientation, scale, and local translation.

The representational properties of early layers in the CNN mentioned above are similar to the response properties of neurons in primary visual cortex (V1), which is the first cortical area in the visual hierarchy of the primate's brain. For example, neurons in area V1 detect primary visual features, such as oriented edges, from input images (, 1959, 1962). Each V1 neuron is selective to a particular orientation, meaning that when a stimulus with this orientation is presented, only selective neurons to this orientation respond maximally. Representation learning methods, which use neural networks such as autoencoders and sparse coding schemes, learn to discover visual features similar to the receptive field properties found in V1 ( ). Bio-inspired SNNs also have obvious footprints in representation learning using sparse coding ( ), independent component analysis (ICA) ( ), and an STDP-based autoencoder ( ).

As mentioned earlier, CNNs commonly use V1-like receptive field kernels in early layers to extract features from stimuli by convolving the kernels over the input (e.g. image). Subsequent layers combine the previous layer's kernels to learn increasingly complex and abstract stimulus features. Representation filters (trained or hand-crafted) and STDP learning rules can be used to develop spiking CNNs. A convolutional/pooling layer trained by a local spike-based representation learning algorithm is shown in Fig. 5. Hand-crafted convolutional kernels have been used in the first layer of a number of spiking CNNs that have obtained high classification performance ( ). Difference-of-Gaussian (DoG) is a common hand-crafted filter that is used to extract features in the early layers of SNNs. This choice is bio-motivated to mimic inputs to the mammalian primary visual cortex. The primate visual system is known to have spatio-temporal (space–time) receptive fields where the RF's spatial characteristics evolve in over a brief time after the onset of a presented stimulus ( ). This is the case for instance for direction selective cells. ( ) present a spiking neuromorphic model that may bear some computational relation to this phenomenon. They use so-called time surfaces acquired by hierarchical incremental clustering to create a hierarchical model for pattern recognition. An elementary time surface recognizes a local spatio-temporal feature provided by an event-based sensor.

One of the deepest (to date) STDP-trained convolutional architectures with multi-layer learning has used a layer of DoG filters as input layer of an SNN that is followed by more convolutional/pooling layers trained by STDP (2018). This network architecture extracted visual features that were sent to an SVM classifier, yielding high accuracy of 98.4% on MNIST. Later, ( ) introduced dual accumulator neurons to address unfavorable interactions of multi-layer training and lateral inhibition. This produced a fully event-driven system inspired by the above design ( ) which used unsupervised STDP to train all layers simultaneously. To train convolutional filters, layer-wise spiking representation learning approaches have been implemented in recent spiking CNNs ( ). ( ) used SAILnet ( ) to train orientation selective kernels used in the initial layer of a spiking CNN. The convolutional layer in this network is followed by a feature discovery layer equipped with an STDP variant ( ) to extract visual features for classification. (2018b) created a new version of SAILnet to improve its locality in space and time. Implementing stacked convolutional autoencoders ( ) showed further improvement in performance on MNIST (99.05%), which is comparable to the traditional CNNs.

Non-spiking CNNs are trained using the backpropagation algorithm. Recently, backpropagation has also been employed for training spiking CNNs ( ). ( ), using approximations developed in ( ), showed how to build a hierarchical spiking convolutional autoencoder (AE) using backpropagation. The spiking convolutional autoencoder is an important module for enabling the construction of deep spiking CNNs. Their proof-of-concept implementation (SpikeCNN) used two learning layers on the MNIST

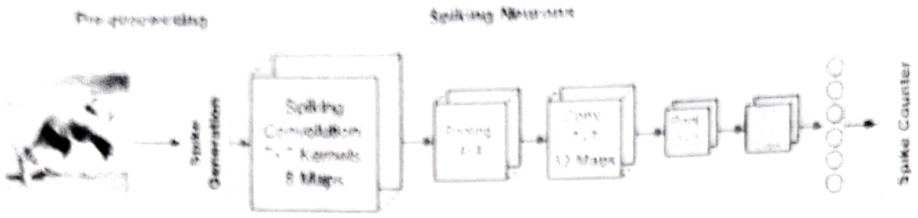

Fig. 6. Spiking CNN architecture developed by [ ]. The input image, after pre-processing, is converted to spike trains based on pixel intensity. The spiking layers use weights trained by a non-spiking CNN. The last component selects the neuron with maximum activity (spike frequency) as the image's class.

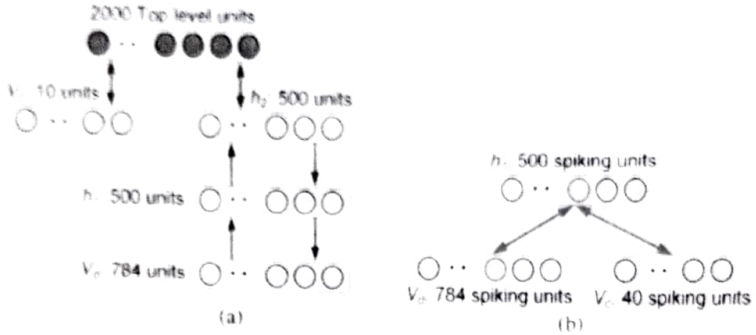

Fig. 7. (a) The DBN proposed by [ ] for MNIST image classification. This network consists of three stacked RBMs with 500, 500, and 2000 representation neurons. The input and output include 784 (as the number of pixels, 28×28) and 10 (as the number of classes, $0, \ldots, 9$) neurons, respectively. (b): The spiking RBM architecture introduced by [ ] consisting of 500 hidden neurons, 784 input neurons, and 40 class neurons (824 visible neurons).

marginalized over the hidden units, both the Hopfield and the HBM systems have been shown to be thermodynamically equivalent. Although Hopfield networks are primarily used as models of pattern association, the thermodynamic equivalence allows them to be simulated by HBMs. In the HBM, the N binary visible units correspond to binary stochastic neurons in the Hopfield network and the P hidden units in the HBM correspond to stored patterns in the Hopfield network. HBMs offer a new way to simulate Hopfield networks while using fewer synapses. Specifically, a Hopfield network requires updating N neurons and $N/(N-1)/2$ synapses, whereas the HBM requires $H+P$ neurons and updating HP synapses, where P is the number of stored patterns. Further developments of this theory are found in [,] and [ ].

3.4. Recurrent SNNs

A neural network is recurrent if its directed graph representation has a cycle. Any network that has a winner(s)-take-all (WTA) module or a softmax module is at least implicitly recurrent because equivalent function is implemented in the brain by mutually recurrent inhibitory connections. Any network trained by backpropagation is also implicitly recurrent because the training algorithm (as explained in Section) presupposes the existence of recurrent connections. Sections and discuss gated SNNs and reservoir models, respectively. Both types of models are intended to process sequential data. The former processes spatiotemporal data and is usually intended as a model of cortical microcircuits. The latter focuses more on sequential data only.

3.4.1. Gated SNNs

Recurrent neural networks (RNNs) are used for processing temporal information. The most common method to train RNNs is backpropagation through time (BPTT), which unrolls the recurrent network for some number of steps into the past, and then trains the unrolled network as if it was a feedforward network. Since the same recurrent weights are shared through all unrolled layers for the resulting feedforward network, there are issues in training for long sequences, specifically the emergence of vanishing and exploding gradients. In the former case, the network stops learning and in the latter, the training becomes unstable (,).

Because of these problems, the research in [ ] introduced an innovation into recurrent networks called a constant error carousel (CEC) that avoided repeated multiplication of derivatives. These have become known as gated recurrent networks (in addition to the CEC, they have trainable gates) and have virtually replaced traditional RNNs. The first gated recurrent network was the long short-term memory (LSTM) (,). LSTMs and other gated recurrent networks (GRUs) (,) are conventional ANNs in the sense that they do not use spiking neurons, but they are also unconventional in the sense that they replace units having recurrent connections with 'cells' that contain state as well as gates and, because of this, can readily adapt to the structure of input sequences. The gates control information flow into, out of, and within the cells. The gates are controlled by trainable weights. The topic we consider in this subsection is the current status of the field with regard to creating spiking LSTMs or gated recurrent networks.

There are only a few recurrent SNNs, in which conventional (non-spiking) RNNs are converted to spiking frameworks. Perhaps the earliest was [ ] which applied the previously discussed conversion method to a (non-gated) Elman network (,) for implementation on neuromorphic hardware. [ ] implemented an energy-efficient spiking LSTM onto the IBM TrueNorth neurosynaptic system platform (,). To do this, they had to solve two problems. The first was to build a spiking LSTM and the second was to implement it on a neuromorphic chip. We focus on the former problem. One of their design choices was to represent positive and negative values using two channels of spike trains. This is bio-plausible and the DoG filters discussed in Section commonly come in two forms for exactly this purpose.

The inputs, outputs, and most of the internal LSTM variables used rate coding. There was one exception to this where the value of the variable representing cell state needed higher precision and was represented by a spike burst code. The main thrust of the paper was overcoming the complications in mapping the LSTM to a neuromorphic chip and accuracy results on standard benchmarks were not reported.

The phased LSTM ( ), although not a spiking LSTM, is well suited to process event-driven, asynchronously sampled data, which is a common task for SNNs. This makes it useful to process inputs from (possibly several) biomorphic input sensors that sample inputs at multiple time scales. It is also potentially useful for processing outputs from SNNs. The innovation of the phased LSTM is the addition of a time gate to the usual set of gates within a memory cell. The time gate is synchronized to a rhythmic oscillation. When 'open', the time gate allows the usual updates to the hidden, output, and cell state vectors. When 'closed' it prevents the updates forcing the hidden and cell vectors to retain their values. The units within these vectors can have separate updates with their own oscillation periods and phases. This allows the phased LSTM to quantize its input at different time scales. In a number of experiments that involved event-driven sampling at varied time scales, the phased LSTM has trained more quickly than a regular LSTM while performing as accurately as the regular LSTM.

3.4.2. *Liquid state machines and reservoirs*

The neocortex, unique to mammals, has the ability to drastically scale its surface area from about 1 square cm in the mouse to about 2500 square cm in the human while keeping its thickness fairly constant (≤ 3 mm). To support this expansion, one hypothesis is that mammals discovered a structural motif that may be replicated and functionally adapted to new tasks. Initially this was called a minicolumn consisting of about 300 excitatory and inhibitory recurrently connected neurons that span the six layers of the three-millimeter-thick neocortex. More recently the term canonical or cortical microcircuit has been used. There has been great interest in modeling this hypothesized module because of the potential insights it may offer to universal computation.

In an effort to model the computations that might be taking place within the canonical neocortical microcircuit, the liquid state machine (LSM) was introduced (,) which has since been partly absorbed by the field of reservoir computing (, ; ,). In an LSM context, a neural reservoir is a sparsely connected recurrent SNN composed of excitatory and inhibitory neurons designed to have enough structure to create a universal analog fading memory. This module is constructed so that it can transform a set of possibly multi-modal spike trains into a spatiotemporal representation whose instantaneous state can be recognized and readout by a layer of linear units.

A reservoir model has three parts:

1. It needs one or more sensory-based, time-varying input streams of spikes or continuous inputs that can be transduced into spikes.
2. It needs a recurrent SNN known as a reservoir or liquid whose synapses may (or may not) be able to learn. The neurons are given physical locations in space as a means to establish connection probabilities, which generally decrease exponentially with distance. Some ideas on why this might be useful are given in ( ). Connections tend to be sparse to avoid chaotic dynamics. The ratio of excitatory to inhibitory neurons is usually about 80% to 20% to reflect the ratio found in the neocortex.
3. It needs (usually) linear readout units which can be trained to recognize instantaneous patterns within the liquid. Part of the motivation for the linearity is to prove that the information in the reservoir's dynamically evolving state can be easily read out.

The neuromorphic cube (NewCube) (, ; , ; ,) is a broad and futuristic model proposed as a unifying computational architecture for modeling multi-modality spatiotemporal data, most especially data related to the brain, such as EEG analysis (,). The core of the NeuCube architecture is a 3D reservoir of spiking neurons trained by an STDP-like mechanism. The reservoir's structure is intended to directly reflect the inter-area connectivity of the human neocortex as revealed by structural connectivity based on anatomy such as diffusion tensor imaging (DTI) and functional connectivity based on measures like functional magnetic resonance imaging (fMRI). This is in contrast to the use of a reservoir as a model of a neocortical microcircuit. The claim of NeuCube is that the connectivity of the brain at the macro scale obeys the properties of a reservoir.

The previous subsection discussed the status of attempts to create spiking versions of LSTMs. Rather than pursuing a direct approach to structurally translating an LSTM to a spiking version, the work of ( ) took a reservoir inspired approach. Their LSNN architecture (long short-term memory SNNs) consisted of four modules labeled: *X*, *R*, *A*, and *Y*. *X* provided multiple streams of input spikes, *R* was the reservoir consisting of excitatory and inhibitory neurons, *A* was a module of excitatory neurons (connected to *X*, *R*, and *Y*) with adaptive thresholds whose purpose was in part to maintain a tight excitatory–inhibitory balance in *R*, and module *Y* consisted of the readout neurons. The LSNN was trained using BPTT (Section) with pseudo derivatives using the membrane potential (as explained in Section). The network achieved comparable accuracy performance to LSTMs on the sequential MNIST benchmark and also on the TIMIT Acoustic-Phonetic Continuous Speech Corpus. Sequential MNIST is a sequence learning benchmark for assessing recurrent network performance first described in ( ). The task is to recognize MNIST digits but now the input is the set of $784 = 28^2$ input pixels delivered sequentially over consecutive time steps. Although the LSNN architecture does not have a direct mapping to the architecture of an LSTM, it has learning abilities that are apparently unique to LSTMs. It has been shown that LSTMs "can learn nonlinear functions from a teacher without modifying their weights, and using their short-term memory instead". In ( ), it was shown that LSTMs had this property.

Another recent study, (,), has attempted to adapt the architecture of a conventional LSTM to be a plausible model of a cortical microcircuit. There were two innovations in this model. First, to facilitate mapping to a microcircuit, the multiplicative gating operations within a standard LSTM were replaced with subtractive operations that could be implemented by lateral inhibitory circuits in the neocortex. This leads to the name subLSTM (subtractive LSTM). Second, to facilitate learning and study using readily available deep learning frameworks, the neural coding scheme was assumed to use rate-coded LIF neurons. This allowed them to model spiking neurons using a continuous approximation that was compatible with deep learning environments. Their bio-plausible subLSTM achieved comparable performance to a standard LSTM on the sequential MNIST task. Similar results were obtained in a language processing benchmark. The subLSTMs did not perform better than standard LSTMs but they have opened a venue for interdisciplinary