ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

Efficient Learning in Spiking Models

Alexander Rast[1], Mario Antoine Aoun[2], Eleni Elia[1] and Nigel Crook[1]

1- Oxford Brookes University - School of Engineering, Computing, & Mathematics
Wheatley Campus, Wheatley, Oxford OX33 1HX - United Kingdom

2- Montreal, QC, Canada

Abstract. Spiking neural networks (SNNs) form a large class of neural models distinct from ‘classical’ continuous-valued networks such as multilayer perceptrons (MLPs). With event-driven dynamics and a continuous-time model, in contrast to the discrete-time model of their classical counterparts, they offer interesting advantages in representational capacity and energy consumption. However, developing models of learning for SNNs has historically proven challenging: as continuous-time systems, their dynamics are much more complex and they cannot benefit from the strong theoretical developments in MLPs such as convergence proofs and optimal gradient descent. Nor do they gain automatically from algorithmic improvements that have produced efficient matrix inversion and batch training methods. Research has focussed largely on the most extensively studied learning mechanisms in SNNs: spike-timing-dependent plasticity (STDP). Although there has been progress here, there are also notable pathologies that have often been solved with a variety of ad-hoc techniques. A relatively recent interesting development is attempts to map classical convolutional neural networks to spiking implementations, but these may not leverage all the claimed advantages of spiking. This tutorial overview looks at existing techniques for learning in SNNs and offers some thoughts for future directions.

1 Spiking neural models: what and why?

A neural network of any type is a model: a mathematical approximation of some complex system. But because the system in question can achieve useful computation, neural models have always had a dual rôle: for computing systems and for biological modelling. These 2 different starting points have led to 2 contrasting model architectures. Computer scientists developed abstract models that act as discrete-time numeric function approximators: ‘classical’ neural networks such as multilayer perceptrons, which were later applied to high-level biological modelling. Computational neuroscientists, meanwhile, worked with continuous-time differential systems: ‘spiking’ neural networks which subsequently generated interest amongst the computer scientists. Spiking networks encode information in transient impulses or events - spikes - that propagate through the continuous-time system asynchronously. Given their origins, by design, such networks can more faithfully approximate real biological dynamics, and thus they are a natural fit for computational neuroscience, whether the goal is to understand the actual function of the brain or to derive computational models that leverage the known advantages of biology-like computing. It is known that because spikes can encode 2 dimensions of information: frequency and phase, on the same signal, they have higher computational capability than classical neural networks at the same level of network complexity. So there are compelling reasons to look at spiking networks. But like any other neural network, to be useful a spiking network needs to be able to learn, and this introduces the question of how does (or can) a spiking network learn improved internal representations or computational capabilities from inputs? This tutorial provides an outline of approaches and some idea of what the relevant decisions are.

2 Model neurons

Before getting into learning mechanisms and rules as such, it is necessary to discuss briefly the various 'flavours' of model neuron. The starkly contrasting goals of different network models have important consequences for neuron model choice.

Leaky Integrate-and-Fire The LIF model is the oldest, simplest, and most widely used of all models, using a single relaxation term. A major advantage is that unlike most other models, it can be solved in closed form.

Adaptive Exponential Integrate-and-Fire The so-called AdEx model is also popular, being relatively simple and offering richer dynamics such as intrinsic bursting and resonant neurons, using an adaptive threshold.

Izhikevich This is a quadratic differential model with rich dynamics, yet a simple computational form. Naïvely implemented, it can lead to instabilities, but good implementations are particularly suitable for temporally-dependent computing. The model is, however, less popular amongst neuroscientists.

Hodgkin-Huxley The H-H model accurately reproduces the biophysics of neurons, at the cost of introducing complex nonlinear ODEs that require accurate solvers. Although popular amongst neuroscientists, it has seen few computational uses, given the existence of simpler models that can reproduce similar dynamics.

Spike-Response Model The SRM is one of a class of extremely simple models for computational applications. An SRM neuron is basically a spike counter which outputs when the count reaches some value. This makes for an almost trivial digital implementation, the major disadvantage being no real dynamics.

Although model choice can become a vexed issue, here, we summarise the 'established wisdom'. The LIF model is the model of choice unless it is known that specific temporal effects like bursting and resonance are necessary. If such effects are needed, computational applications must consider AdEx or Izhikevich but beware of stability considerations. Neurobiological applications may demand Hodgkin-Huxley for rigour, but this is only crucial if the low-level neuronal dynamics are the subject of enquiry. We now turn to the question of model synapses, which is the basis for learning.

3 Model synapses

Synapses, in virtually all neural networks, have 2 distinct roles: they perform a computation on the input to convert an input spike event to an output current injection, typically by some scaling that corresponds to a 'weight'; and they also realise a learning rule that modifies the conversion itself. They may attempt to be biologically realistic or not: the former implement, like neurons, some real temporal dynamics; the latter may be a simple discrete output value or counter. Synapses are typically either excitatory, in fast ('AMPA') or slow ('NMDA') variants, or inhibitory, again with fast ('GABA-A') and slow ('GABA-B') types. Synapses are rarely bipolar, except for some computational models that discard biological realism.

Extremely generally, synapses result in an excitatory post-synaptic potential (EPSP) of the following form, taken from [1]: $\mathbf{EPSP_n} = \mathbf{A_{se}R_n u_n}$, where index n indicates the nth incident spike, A_{se} is the 'absolute' synaptic efficiency or weight, R_n is the total available 'resources' and u_n is the transmission dynamics. All major classes of biological synapse, however, follow *transmission* dynamics (the 'channel kinetics') that feature an abrupt rise at spike onset to some high output, and a comparatively slower relaxation (often exponential) to their resting equilibrium. These are modelled typically either using synaptic conductances, solving some system of ODEs much like neurons ('conductance-based' synapses) or various forms of simple exponential rules - often with a pair of time constants for onset and relaxation - ('current-based' synapses).

Given the general form of a neuron model $c\frac{dV}{dt} = -i_m + i_s$, where V is the membrane voltage, c the membrane capacitance, i_m the membrane current and i_s the synaptic current, a typical conductance-based model has the form

$$\left.\begin{aligned} \frac{dP}{dt} &= R_n(1-P) - \beta_g P, \\ i_s &= \frac{1}{c} g_s P(V - E_s) \end{aligned}\right\} \qquad \text{Conductance-based}$$

where P is the synapse open channel probability, g_s the fully-open synaptic conductance, and E_s is the synaptic reversal potential. β_g is a constant. It can be seen that $A_{se} = \frac{1}{c}g_s$.
A typical current-based model has the form

$$\frac{di_s}{dt} = R_n(A_{se} - i_s) - \beta_i i_s \qquad \text{Current-based}$$

where again, β_i is a constant. Learning (at this level usually called *plasticity*) usually corresponds to some update to either the A_{se} term ('ionotropic') which updates the weight, or R_n ('metabotropic') which updates a proxy variable that describes the surrounding resource availability shared between nearby synapses. In typical synapses R_n is at its resting value when the synapse is inactive, dropping quickly to 0 when the synapse is fully open, and recovering slowly with some time constant larger than $\frac{1}{\beta}$. Occasionally, computational learning models may update u_n, but as noted, this is not typical in neurobiological models.

3.1 Plasticity mechanisms

Most plasticity rules use some form of Hebbian [2] learning dependent on pre/ post-synaptic spike pairings. Gerstner, et al.[3] introduce one of the earliest such models. But overwhelmingly the most well-known, well-used, and well-studied plasticity rule is spike-timing-dependent plasticity (STDP)[4][5], an 'ionotropic' Hebbian rule deriving from the relative timing of pre- and post-synaptic spikes. The original 'spike-pair' rule has spawned several further variants - 'triplet' [6] and even 'quadruplet' STDP which attempt to account for more context. However, the method of computing the magnitude of the update, once a synapse is eligible for modification, leads to particularities in the dynamics.

So-called 'additive' STDP [7] leads to a distinct bistable distribution of weights, which can be useful in 'competitive' networks such as a winner-take-all circuit, but creates problems in other types, e.g. convolutional-style networks, where high expressivity is needed. 'Multiplicative' STDP [8], instead of updating the weight by a fixed quantity, scales the update proportional to the current weight of the synapse, largely solving issues with bistability. Multiplicative STDP, however, can be sensitive to data order and distribution and does not encourage synapses to specialise to a given input. The result can be a 'pool' of undifferentiated synapses [9].

3.2 Learning rules

At a *network* level, plasticity mechanisms drive larger-scale learning rules, which follow the traditional division between supervised and unsupervised rules. Fundamentally, Hebbian mechanisms are unsupervised learning rules by themselves, but they can also be used in supervised contexts by providing a strong driving input (and there is evidence e.g. in Purkinje cerebellar neurons [10] that such supervised methods occur in biology as well). Numerous specific algorithms have been proposed. Table 1 lists some important and influential SNN learning rules.

3.2.1 Supervised rules

One early trend in the development of *supervised* learning rules for SNNs was a series of attempts to map the existing backpropagation [19] algorithms designed for continuous-valued (nonspiking) neurons to spiking networks. A simple approach is to transfer the weights directly from a trained nonspiking network to an equivalent rate-coded spiking representation [20]. Whilst such a method may not really represent a true spiking learning algorithm, progress has since been made in mapping backpropagation-like algorithms explicitly to spiking networks using both rate [21] and temporal [11] coding, though these are unlikely to be biologically plausible.

A very different direction is represented by models that use a 'training' signal, essentially an auxiliary input to the network with its own spike pattern. Such models typically rely on coincident firings of the training signal and the input to potentiate or depress synapses according to STDP-like mechanisms. Arguably

Learning Algorithm	*Year*	*Approach*
Supervised		
SpikeProp [11]	2002	Back Propagation
ReSuMe [12]	2005	STDP & Delta Rule
SPAN [13]	2013	Delta Rule and Kernel
FOLLOW [14]	2017	Local learning combines postsynaptic error and presynaptic activity
EventProp [15]	2021	Gradient based learning allowing backpropagation through discrete spike events
Unsupervised		
Liquid State Machine [16]	2002	Single-layer learning with random 'reservoir'. Amenable to supervised training
Winner-Take-All [17]	2015	Competitive learning in groups
Stable STDP [18]	2020	2-factor, multiplicative STDP

Table 1: Historical and Fundamental Spiking Learning algorithms

the most well-known of these approaches is the ReSuMe [12] model, which uses a single training input to learn specific sequences by generating an input when a spike 'should' have occurred. Though not unusually *efficient*, accuracy has been improved in successive models [22].

3.2.2 Unsupervised rules

An interesting idea recently explored was the use of an unsupervised learning step to 'pre-train' a convolutional spiking model which was then 'fine-tuned' using conventional backpropagation [23]. However, the majority of unsupervised rules have been based on competitive or chaotic dynamics. Such properties have been used in 'reservoir' computing models (of which the Liquid State Machine [16] is the most well-known example) to yield competing delay paths, and in winner-take-all circuits [17] where individual neurons compete. These rules exploit phase transitions in the underlying dynamics to encourage individual neurons or groups to compete for stable regions in state space.

3.2.3 Nonlinear Neural Dynamics

Such phase transitions, and the role of dynamics in the overall function of the network, is the focus of another area of investigation: Nonlinear Neural Dynamics. Two of the most common roles found in the literature are communication and representation. Weakly connected chaotic systems have a strong propensity to synchronise with each other. Synchronisation has been studied between individual cells and between brain networks [24]. In these cases, learning is often facilitated by modifying the connection weights between cells and/or subnetworks that are exhibiting similar dynamic behaviours.

The properties of chaos have also been used to explore the representation of

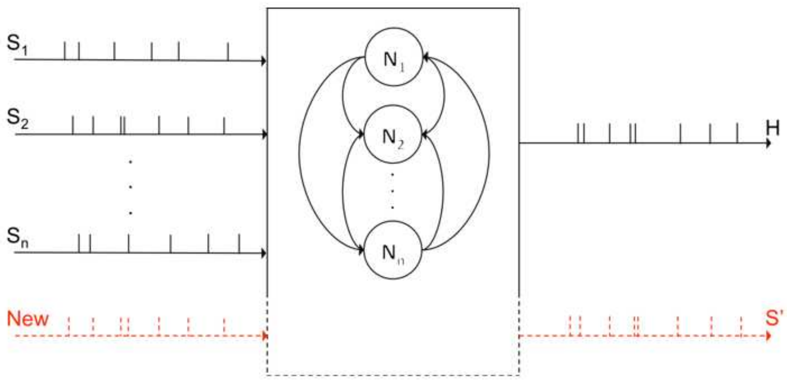

Fig. 1: Chaotic Liquid State Machine. Multiple spiking inputs (S_1 to S_n) are to be encoded by a pool of neurons (N_1 to N_n). These synchronise to a single output (H) which encodes the inputs in a saved spike train. When a New input arrives (dashed), the pool generates a corresponding output (S') that is compared to the Hypothesis (H) using a similarity measure.

state or information in neural systems. Two general approaches have been taken. One is to use the rich nonlinear dynamics of chaos to search a solution space or amplify the differences between input signals [25]. Here the learning takes place in the connection weights between the chaotic network and a 'readout' layer of simple linear neurons. The second approach is to model chaotic systems that can be easily controlled or 'nudged' into Unstable Periodic Orbits (UPOs) [26], or attractor sub-spaces [27]. These, in turn represent context-specific input responses that can be learnt by weight changes that affect the parameters of the chaotic system - a combination of communication and representation. An example is the Chaotic Liquid State Machine [28] (Fig 1).

4 Considerations and Recommendations

As has been seen, spiking neural networks are used in 2 distinct domains: for biological modelling and for computing applications. There is also a third, cross-disciplinary domain, that of using networks in a practical application (e.g., controlling an autonomous robot) with the expectation that it may inform understanding of comparable mechanisms in biology.

Modern computational neuroscience is typically interested in modelling at large scale, where computational efficiency is critical, but there is typically no requirement that the *computation* itself perform anything useful. High-level learning rules are immaterial, and the focus in on the choice of plasticity mechanism. Thus model choice, as a practical matter, can be distilled to one basic recommendation: *choose the simplest model you can that meets the goals of the*

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

study. In many cases, this can simply be 'vanilla' additive or multiplicative STDP. Only where the exact kinetic of the plasticity rule itself is the subject of investigation are more complex dynamics typically appropriate. These will, naturally, vary according to the hypotheses of the study itself. Similar considerations apply to the neuron model, and it is unsurprising that the majority of large-scale studies have used LIF neurons (in spite of known limitations).

In the cross-domain case, much has been achieved with straightforward LIF-STDP combinations; where there is design variability, it tends to have come in the neuron model, with adaptive-exponential, Izhikevich, and indeed Hodgkin-Huxley models all having been attempted. Many approaches have also employed dynamic global parameters, as proxies for neuromodulatory processes, and these have occasionally yielded interesting further network behaviours, although the impact on task behaviour has been more difficult to assess. Curiously little work, however, has been done comparing different pairings of synaptic model with neuron model; this is clearly a rich field for future studies.

In the pure computational domain, although significant improvements in learning performance have been achieved over time, spiking neural networks as yet have not been able to consistently outperform conventional (graded-response) neural networks when it comes to accuracy. One may then wonder, what is the use of spiking networks for real applications? The customary response of most works has been 'accuracy is not the only metric of interest', and some obvious cases can be enumerated and recommendations given about learning rules.

Spatiotemporal Applications Where data has spatiotemporal characteristics (e.g. video scene segmentation), the learning rule needs to be able to capture phase-frequency relationships. Additive STDP is typically unsuitable. Successful models often use global neuromodulatory parameters to yield good results. An intriguing area for future work is in merging the ideas from complex-valued neural networks with spiking learning [29].

Power-Limited Embedded Systems Obviously, in cases where power efficiency is paramount (e.g. microdrones), the learning rule and plasticity model should be as simple as possible whilst permitting fast learning; WTA networks with additive STDP are an example of an appropriate choice.

Real-Time Systems In a real-time context, where instant response is crucial, what is needed is the fastest synaptic update possible. Typically this will be done asynchronously, thus the learning rule needs to be continuous rather than batch-driven. Unsupervised methods are much simpler to apply, and an underlying plasticity rule like spike-pair rule (using the fewest number of pairings to compute the update), is likely to yield highest-performance results.

Neuromorphic Hardware Platforms When the target hardware is itself a 'neuromorphic' chip [30], specifically designed and optimised to implement spiking networks, large efficiency gains may be possible, typically with some constraints on synaptic model choice. Implementing a new model on-chip may require specialist familiarity with the device; for programmable chips, the most practical recommendation is: use the *oldest* learning rule (i.e. the one first implemented on-chip). It is likely to be the most mature, and may be included

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

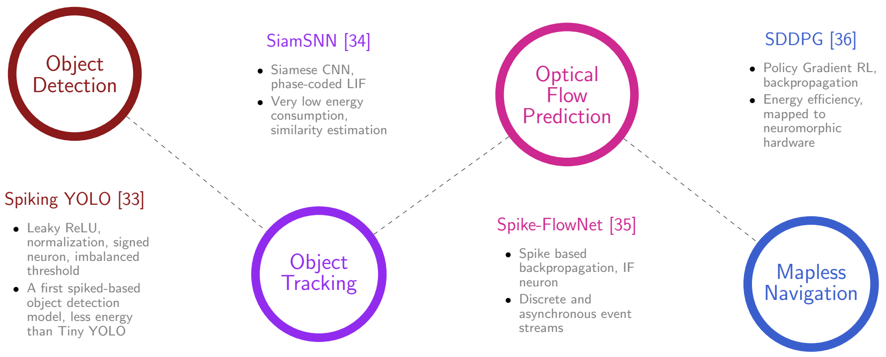
Object Detection
SiamSNN [34]
• Siamese CNN, phase-coded LIF
• Very low energy consumption, similarity estimation
Optical Flow Prediction
SDDPG [36]
• Policy Gradient RL, backpropagation
• Energy efficiency, mapped to neuromorphic hardware
Spiking YOLO [33]
• Leaky ReLU, normalization, signed neuron, imbalanced threshold
• A first spiked-based object detection model, less energy than Tiny YOLO
Object Tracking
Spike-FlowNet [35]
• Spike based backpropagation, IF neuron
• Discrete and asynchronous event streams
Mapless Navigation

Fig. 2: Recent Energy-Efficient SNN algorithms (Selected from [32])

in standard libraries.

The 'Red AI' relation [31] introduces a simple metric for energy efficiency:

$$Cost(R) \propto E \cdot D \cdot H \tag{1}$$

where $Cost(R)$ is the total (energy) cost, E is the (mean) energy cost per example, D is the size of the dataset and H is the number of hyperparameters (assuming sweeps will be made in the course of training over different tuning hyperparameters). Algorithms which reduce any of factors E, D or H can thus yield large improvements in overall cost. The presence of the H term indicates the importance of formal methods for hyperparameter estimation, as a future research topic. More generally, we may estimate for the total energy cost as:

$$\tilde{E}_t = D_S N_w (\bar{N}_u E_u \bar{F}_u) \tag{2}$$

where $\tilde{E}_t$ is the total cost, D_S is the size of the state space, N_w the number of weights in the network, $\bar{N}_u$ the mean number of atomic weight updates per state-space change, E_u is the energy per atomic (weight) update, and $\bar{F}_u$ is the mean fraction of weights updated per state-space change. Most efficient spiking implementations concentrate on $\bar{N}_u$ and E_u since these are typically the most subject to algorithmic variation. Fig. 2 gives some recent SNN algorithms which, according to [32], are claimed to be energy efficient by their authors.

5 Conclusions

Surveying the 'landscape' of efficient learning for spiking networks, it is clear for the moment that supervised methods that attempt to leverage the gains of 'classical' machine learning techniques still prevail, despite evidence that such approaches may not be energy efficient. There is clearly some gain to be had in reducing reliance on hyperparameter sweeps through better formal methods, and further gains in developing learning algorithms that quickly converge on near-optimum solutions, reducing the crucial $\bar{N}_u$ term. Equally, however, unsupervised methods remain underdeveloped, or perhaps it should be said that efforts have produced a kaleidoscope of different ad-hoc techniques with relatively minimal formal theory. Again, the need for stronger formal theoretical development is very clear. Fundamentally, however, biological learning is effective, even in large state spaces, with far smaller *de facto* datasets than modern deep learning networks need, strongly suggesting the existence of learning rules with orders-of-magnitude better $\bar{N}_u$ performance. Methods based on control of noisy chaotic systems hold significant promise here, and deserve further exploration. In the interim, however, use of supervised methods in power-constrained or real-time systems, where modern deep learning methods are poorly suited, is to be encouraged, such as control of autonomous flying drones. Overcoming problems in such practical applications could be a spur to future theoretical development, potentially alongside computational neurobiology experiments, yielding advances that help to 'crack' the larger problem of efficient learning. Much needs to be understood, but the existence proof is out there, it only awaits further insights, some of which are sure to come from research in spiking neural networks.

References

[1] H. Planert et al. Dynamics of Synaptic Transmission between Fast Spiking Interneurons and Striatal Projection Neurons of the Direct and Indirect Pathways. *J. Neurosci.*, 30(9):3499–3507, 2010.

[2] D. O. Hebb. *The Organization of Behavior*, chapter 4, pages 60–78. Wiley, New York, NY, 1949.

[3] W. Gerstner et al. A Neuronal Learning Rule for Sub-millisecond Temporal Coding. *Nature*, 383(6595):76–78, 1996.

[4] C.Q. Bi and M.M. Poo. Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. *J. Neurosci.*, 18(24):10464–10472, 1998.

[5] H. Markram and P. Tsodyks. Redistribution of Synaptic Efficacy Between Neocortical Pyramidal Neurons. *Nature*, 382(6594):807–810, 1996.

[6] J.-P. Pfister and W. Gerstner. Triplets of Spikes in a Model of Spike-Timing Dependent Plasticity. *J. Neurosci.*, 26(38):9673–9682, 2006.

[7] S. Song, K. D. Miller, and L. F. Abbott. Competitive Hebbian learning through spike-timing dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2010.

[8] M. C. W. van Rossum, G. Q. Bi, and G. G. Turrigiano. Stable Hebbian Learning from Spike Timing-Dependent Plasticity. *J. Neurosci.*, 20(23):8812–8821, 2000.

[9] M. Gilson and T. Fukai. Stability versus Neuronal Specialization for STDP:Long-Tail Weight Distributions Solve the Dilemma. *PLoS One*, 6(10), 2011.

[10] D. Jaeger and J. M. Bower. Synaptic Control of Spiking in Cerebellar Purkinje Cells: Dynamic Current Clamp Based on Model Conductances. *J. Neurosci.*, 19(14), 1999.

[11] S.M. Bohte, J.N. Kok, and H. La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[12] F. Ponulak and A. Kasiński. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Comput.*, 22(2):467–510, 2010.

[13] A. Mohemmed et al. Training spiking neural networks to associate spatio-temporal input–output spike patterns. *Neurocomputing*, 107:3–10, 2013.

[14] A. Gilra and W. Gerstner. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *eLife*, 6:e28295, 2017.

[15] T.C. Wunderlich and C. Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):12829, 2021.

[16] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.*, 14:2531–2560, 2002.

[17] P. Diehl and M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.*, 9:99–113, 2015.

[18] F. Paredes-Vallés, K. Y. W. Scheper, and G. C. H. E. de Croon. Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception. *IEEE T. Patt. Anal. Mach. Intell.*, 42(8):2051–2064, 2020.

[19] D. Rumelhart, G. Hinton, and R. Williams. *Learning internal representations by error propagation*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[20] P. O'Connor et al. Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.*, 7(178), 2013.

[21] J. H. Lee, T. Delbruck, and M. Pfeiffer. Training Deep Spiking Neural Networks using Backpropagation. *Front. Neurosci.*, 10, 2016.

[22] A. Taherkhani et al. A Supervised Learning Algorithm for Learning Precise Timing of Multiple Spikes in Multilayer Spiking Neural Networks. *IEEE T. Neural Netw. Learn. Syst.*, 29(11), 2018.

[23] C. Lee et al. Training Deep Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning. *Front. Neurosci.*, 12, 2018.

[24] S.P. Kuo, G.W. Schwartz, and F. Rieke. Nonlinear spatiotemporal integration by electrical and chemical synapses in the retina. *Neuron*, 90(2):320–332, 2016.

[25] R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural networks*, 20(3):323–334, 2007.

[26] N. Crook. Nonlinear transient computation. *Neurocomputing*, 70(7-9):1167–1176, 2007.

[27] K. Aihara, T. Takabe, and M. Toyoda. Chaotic neural networks. *Phys. Lett. A*, 144(6-7):333–340, 1990.

[28] M.A. Aoun and M. Boukadoum. Chaotic liquid state machine. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 9(4):1–20, 2015.

[29] A. Baranski and T. Froese. Efficient Spike Timing Dependent Plasticity Rule for Complex-Valued Neurons. In *Proc. AILIFE 2021: 2021 Conf. Artific. Life*, 2021.

[30] C. D. Schuman et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Comput. Sci.*, 2:10–19, 2022.

[31] R. Schwartz, J. Dodge, N.A. Smith, and O. Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

[32] K. Yamazaki et al. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7):863, 2022.

[33] S. Kim et al. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11270–11277, 2020.

[34] Y. Luo et al. SiamSNN: Siamese Spiking Neural Networks for Energy-Efficient Object Tracking. In *Int. Conf. Artific. Neur. Netw.*, pages 182–194. Springer, 2021.

[35] C. Lee et al. Spike-FlowNet: Event-Based Optical Flow Estimation with Energy-Efficient Hybrid Neural Networks. In *European Conference on Computer Vision*, pages 366–382. Springer, 2020.

[36] G. Tang, N. Kumar, and K.P. Michmizos. Reinforcement co-Learning of Deep and Spiking Neural Networks for Energy-Efficient Mapless Navigation with Neuromorphic Hardware. In *2020 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 6090–6097. IEEE, 2020.