Expanding Spiking Neural Networks With Dendrites for Deep Learning

Mark Plagge
Sandia National Laboratories
mplagge@sandia.gov

Suma George Cardwell
Sandia National Laboratories
sgcardw@sandia.gov

Frances S. Chance
Sandia National Laboratories
fschanc@sandia.gov

Abstract

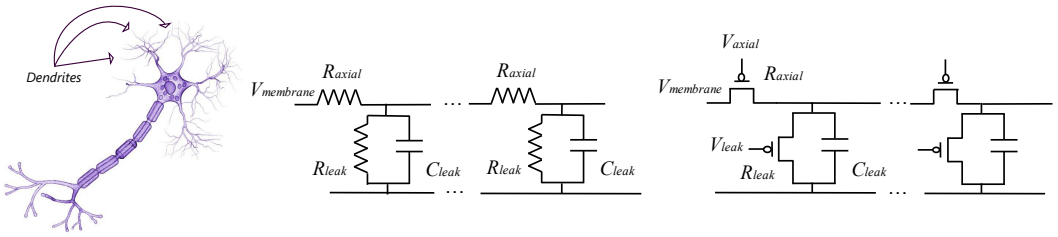
As deep learning networks increase in size and performance, so do associated computational costs, approaching prohibitive levels. Dendrites offer powerful nonlinear "on-the-wire" computational capabilities, increasing the expressivity of the point neuron while preserving many of the advantages of SNNs. We seek to demonstrate the potential of dendritic computations by combining them with the low-power event-driven computation of Spiking Neural Networks (SNNs) for deep learning applications. To this end, we have developed a library that adds dendritic computation to SNNs within the PyTorch framework, enabling complex deep learning networks that still retain the low power advantages of SNNs. Our library leverages a dendrite CMOS hardware model to inform the software model, which enables nonlinear computation integrated with snnTorch at scale. Finally, we discuss potential deep learning applications in the context of current state-of-the-art deep learning methods and energy-efficient neuromorphic hardware.

1 Introduction

Even as deep learning and large-scale neural networks are rapidly transforming the space of computing, their implementations are still limited by the end of Dennard scaling and limits of hardware performance scaling. Spiking Neural Network (SNN)s are a category of brain-inspired machine learning algorithms that leverage event-driven binary-communication connectivity to address low-energy data transfer, one of the largest energy consumption areas in deep learning [27]. In spite of offering one to two orders of magnitude improvement in energy efficiency [27, 18], these networks generally use simple Leaky Integrate-and-Fire (LIF) neuron models which have limited expressivity; SNNs using LIF neurons tend to perform worse than SOTA deep learning models.

In biological systems, dendrites are a structure that enable *pre-processing* of inputs en route to the soma, providing much of the neuron's overall computational capabilities [14]. Dendrites perform computations that include non-linear filtering, coincidence detection, directional selectivity, and amplification of synaptic inputs as key "compute-on-wire" properties of the "dendritic toolkit" [21, 36]. We hypothesize that implementing dendrites in hardware and utilizing them as key computational elements in our algorithms will provide increased computational capabilities while increasing overall efficiency. As an example, [25, 10] shows a demonstration of hardware dendrites computing simple functions while transferring data from input to the output soma, and [6] showcases some of the more advanced computational capabilities of dendrites in the context of spatial and temporal pattern recognition. Thus, dendrites implemented in hardware offer an approach to maintaining the efficiency of a simple LIF based SNN while matching the machine learning performance of traditional ANNs. The primary challenges to realizing dendrite-enabled neural networks include a lack of tools focused on large-scale machine learning as well as a lack of hardware support that leverages the energy efficiency of these networks.

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(MLNPCP 2023).



(a) (b) (c)

Figure 1: Dendrites and hardware models of dendrites. In (a), we show a neuron schematic with labeled dendrites. Biological neurons can perform a wide array of sub-threshold computation (pre-threshold) through dendritic processing, integration, and synaptic weighting of inputs. In (b), we show a resistor-capacitor circuit (RC circuit) to model a passive dendritic cable. In (c), we show an example of a CMOS transistor-based model to emulate the RC circuit shown in (b) in silicon. This sub-threshold CMOS dendrite model is the basis for our dendrite model. The leak and axial conductance of the transistors can be modulated using the gate of the transistor.

Increasing adoption and usage of SNNs with dendritic components requires a rich set of tools and components that machine learning domain experts can use to build out and evaluate large-scale networks of dendrite-enabled neurons. Of the existing SNN deep-learning tools available, no current offerings provide a deep-learning oriented multi-compartment dendritic model [15, 22, 24]. Our work builds on PyTorch with SpykeTorch [24] to create a deep-learning-oriented dendritic library. In this work, we will discuss the implementation of a multi-compartment dendrite library, demonstrate how dendrites can reduce the number of LIF neurons required to learn functions and reduce the number of signals passed over the SNN, and future applications of the dendrite-enabled LIF neuron. We envision that this library may provide a touchstone for further development of deep SNNs, which could justify new development of high-efficiency dendrite enabled SNN hardware designs in the future.

2 Related work

2.1 Dendrites in hardware designs

Figure 1 illustrates a linear passive cable model that represents a dendritic cable. As seen in Figure 1(c), two transistors can represent a single stage in this dendrite. This lightweight dendrite component, consisting of two transistors, provides a low energy and high density dendrite hardware implementation, paving the way for highly dense dendritic hardware networks.

Dendrites have been modeled in silicon in analog and mixed-signal CMOS chips [29, 33, 38, 3, 17]. These *in silico* neuromorphic systems not only model synaptic memory along the dendrite cable, but also model key ion-channel properties to capture dynamics observed in dendrites. There is also interest in leveraging beyond-CMOS dendrites to replicate dendrite-like density and connectivity [20, 19, 4]. These devices provide energy-efficient computation through this re-creation of dendritic structures. This hardware structure provides computation on the wire, as the computation and data transfer occur on the same circuit.

2.2 Dendrites in spiking neural networks for machine learning

In terms of machine learning, the addition of dendrites have been shown to provide dynamic behavior as well as increase the computational power of a spiking neuron [1, 34, 37]. Dendrites in a machine learning network have been leveraged to provide filtering of input signals, logical operations, amplification of signals, coincidence detection, and other dynamic computational operations [1, 2, 7], with some research showing a combination of dendrites and synaptic plasticity can recreate error propagation [32]. These applications show a wide potential in the realm of machine learning applications for spiking networks [41]. Adding dendrites to a spiking neural network, in this domain, adds parameters to the network and can be conceptually seen as adding layers to a non-spiking neural network. These properties of dendrites increase the expressivity of point neuron models and show

interesting properties that have the potential to dramatically improve the performance of spiking neural networks.

Adding dendrites to a spiking neuron: Using dendrites to inform or enhance SNN design has resulted in improvements in network performance [39], online-learning and feature updates [13, 12], and an implementation of novel activation functions [11] that enables a single neuron to classify the XOR logical function. Further research has also shown that dendritic SNNs can help improve online training [30, 31], keyword detection [10], along with several other categories of applications [1, 16, 42].

2.3 Spiking models and software

There is a wide variety of biological neuron modeling tools, SNN modeling tools, and other software frameworks available and in development. These range from highly detailed simulations of biological processes, "middle-ground" simulations which allow larger scale network design, and simplified models. There is, of course, some overlap between these categories, but most SNN tools fall within this spectrum.

Some software libraries attempt to bridge the detailed simulation techniques with large scale simulations. Dendrify, for example, provides dendrite models at a fairly large scale with somewhat biologically accurate models [26]. Based on the Brian 2 simulation system [35], Dendrify extends Brian 2's neuron models to include a variety of dendrite behaviors. This tool, however, does not have native support for surrogate gradient descent nor does it support extremely large SNN models, as the underlying Brian 2 system uses numerical solvers for ODEs in the backend. Other tools, such as Norse [28], provide simplified multi-compartment models for modeling synaptic plasticity, synaptic recurrence, and other dynamic components of a spiking neuron. These tools do provide some dendritic benefits, but they do not allow explicit dendrite design within the framework. Our model and tool are based on a hardware implementation of a dendrite cable as described in [25], simplified for reasonable performance, and coupled with snnTorch to provide surrogate gradient descent learning methods.

3 Methods

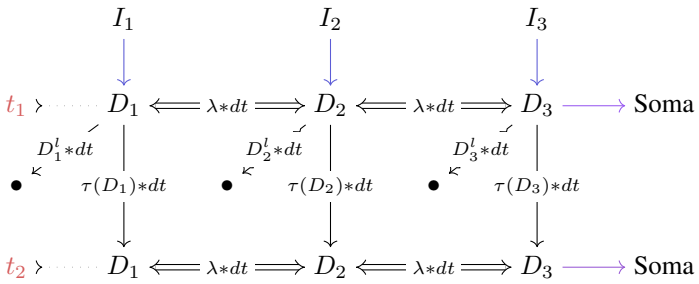


Figure 2: Composition of a three dendrite network. $I_{1...3}$ are the inputs to dendrites $D_{1...3}$. Each dendrite sends a signal in the spatial and temporal dimensions, with a rate determined by λ and τ respectively. There is a leak and input weight component for each dendrite.

Our deep-learning focused dendrite library focuses on a linear cable model based on a hardware dendrite model which would potentially provide high-efficiency computation. Our model uses PyTorch and provides a dendrite building block for SNN development. To emulate the binary spike signaling found in SNNs and enable learning, we leveraged snnTorch [9] which is a machine learning enabled spiking neuron library. This library allows users to create SNNs from within PyTorch, leveraging this well-known machine learning library for spiking networks. In this section, we will discuss the methods used to implement a machine-learning focused dendrite library.

snnTorch provides spiking neuron functions which compute state, with either an internal state storage option or as a parameter to the forward function. In our library, dendrites are modeled as stateful components which save the previous dendrite current values computed from the previous time-step.

We implement the temporal and spatial current spread as a set of iterative computations which compute the change in voltages based on a time value. Each dendrite has a definable temporal modifier, a spatial value, a set of weights, and a configurable exponential kernel. The temporal component, τ, represents the signal delay across taps, while λ represents the spatial signal transmission constant. These values are used to create a kernel which iterates through the number of time-steps specified to compute the new current in the dendrite states. This is a simplification of the more complex active dendrite model. This simplification enables larger-scale computation and is compatible with the surrogate gradient functions provided in the snnTorch library.

Spiking networks in snnTorch use input data in a vector of $M \times N \times I$ size, where M is time, N is batch size, and I is the input size (which can be *n*-dimensional). Our dendrite network implements a resolution parameter, dt, which represents the number of "ticks" that occur between between M, and a t value, which represents the time in-between each T. If dt is one, then one iteration is computed at an input. Smaller values of dt increase the number of iterations over the dendrite during a forward pass. If the dendrite models are used in a traditional PyTorch network that does not consider state or temporal changes of layers over time, the T value enables the dendrites to expand each input tensor into time-steps. For SNNTorch or temporal-aware networks, the T value can be set to one, and the input tensor's time dimension will provide the needed temporal space.

Our library solves a dendrite model that is based on the hardware design in [25]. From the initial cable model, with I as inputs, D_l as per-dendrite leak,τ as the per dendrite connection time constant, and λ as the dendrite space constant, we construct a simplified dendritic computational step.

Taking the multi-step ODE system of equations from [25] as a starting point, rather than solving the ODE for a set number of time steps and resolution, we iterate over each dendrite pair during a forward pass, computing the change in dendrite voltage. Before the dendritic operations, the external inputs to each dendrite are modified by a weight value, such that $I_d = I_d * D_w$. These inputs are applied to the dendrites, and the current in each dendrite is computed using this iterative method. Figure 2 shows this method on a three dendrite network over two time steps, which would represent a dt of 0.5. To simplify the equation, we define:

$$I_d = \text{Input}_{d_n} * \tau * dt.$$

If there are N dendrites in a chain, the first dendrite's computation at step t is computed as

$$D_1^t = (D_1^{t-1} + I_1) + (D_2^{t-1} * \lambda * dt),$$

the inner dendrites, where $n > 1$ and $n < N$ are computed as

$$D_n^t = (D_n^{t-1} + I_n) + (D_{n-1}^{t-1} * \lambda * dt) + (D_{n+1}^{t-1} * \lambda * dt),$$

and the final dendrite's new value is computed as

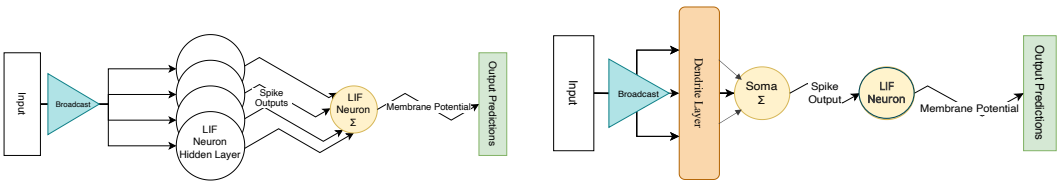
$$D_n^t = (D_n^{t-1} + I_n) + (D_{n-1}^{t-1} * \lambda * dt),$$

for the number of intermediate steps requested. The output of the final dendrite in the chain is sent to an integrate and fire neuron implementation from SNNTorch which represents the soma. This neuron acts as the soma of the overall dendrite neuron. As an alternative, the output value of the last dendrite can be sent to the next module or layer in the network directly, for more advanced or non-spiking network designs.

The end result of this computation is that the dendrite layer behaves as hybrid of a pooling layer, an activation layer, and as a nonlinear transformation layer. This dendrite chain modulates the soma's output behavior, creating a pre-computation unit. In the next section, we demonstrate this behavior against a set of nonlinear functions.

4 Capabilities

To further demonstrate the capabilities of dendrite enabled LIF neurons, we implemented a simple fully-connected neural network using LIF neurons and a similar network using dendrite-enabled neurons. We configured the input layer to broadcast the signal to a hidden layer, consisting of 256 LIF neurons, 16 LIF neurons, or 16 dendrite-enabled LIF neurons. This layer communicates to the output neuron, which sends the membrane potential as output of the network. This network



(a) (b)

Figure 3: Architecture overviews of the networks trained on $\sqrt{x}$ and Mish(x). In (a), the architecture of the simple fully-connected SNN is shown. In (b), the dendrtic network architecture is shown.

is an implementation of spiking network linear regression method, which learns the objective by minimizing the mean square error.

The SNN network is based on a design used as an exemplar model from the snnTorch library. A simplified version of the architecture is shown in Figure 3a. The dendritic version, shown in Figure 3b, replaces the 256 neuron layer with 16 dendrite-enabled neurons, each with 16 dendritic inputs. This effectively takes the 256 output value broadcast from the previous layer, and reduces the output to 16 signals. We chose this particular configuration so that there are still 256 computational operations taking place, but the total data sent to the summary neuron is only 16 wide. In dendritic hardware, the 256 dendrite computations would be computed as the signals are propagated through the network which is leveraging the compute-on-wire capabilities of such hardware. Furthermore, this reduces the total signals that would be sent "off-neuron", which would reduce a network on chip communication cost. The final layer is a single dendritic neuron with a 16 dendrite input chain which combines the hidden layer signals into the single soma. This neuron acts as a non-linear sum of inputs from the previous layer. Finally, the output of this neuron is the membrane potential, rather than the spike train. This follows the linear regression example from the original work. The 16 neuron SNN is the same model as the 256 neuron model, but with far fewer neurons in the hidden layer.

In this example, our target functions are $f\left(x\right) = \sqrt{x}$ and $f\left(x\right) = \text{Mish}\left(x\right)$, which are a pair of simple nonlinear functions. Mish is a self-regularized activation function, defined as $f(x) = x * \tanh(\text{Softplus}(x))$ [23]. Each input x of training data consisted of 1,000 random samples from the range X, where $X = \{x \in \mathbb{R} | x > 0 | x < 1\}$ for $\sqrt{(x)}$, and $X = \{x \in \mathbb{R} | x > -3 | x \leq 1\}$ for the mish function. As spikeTorch and dendrites expect temporal data, the dataset shape became $10 \times 1 \times 1$, with 10 copies of x in each batch.

Neuron weights and dendrite parameters were randomly initialized for the test. In the case of the spiking network, the neurons were set to learn weight, threshold, and reset values. The dendritic neurons were set to learn λ, τ, leak values, as well as the soma's threshold value. We trained each network for 100 iterations over the sampled x values with a batch size of one. The results of these tests are shown in table 1.

Training was done on a workstation computer, running an Intel Xeon E5-1660 V4 with an Nvidia Titan Xp GPU. The 256 neuron SNN took an average of 17.99s per iteration on the dendritic SNN network, and 19.16s per iteration on the 256 sized hidden layer SNN network. The primary reason for this performance is the iteration needed during training. The snnTorch framework uses iterations during training to simulate time-steps. This adds a significant level of overhead to training. There may be techniques to improve this performance, including leveraging more recurrent network layouts.

	256 LIF Hidden Layer	**16 LIF Neuron Hidden Layer**	**16 Dendrite-LIF Hidden Layer**
MSE $\sqrt{x}$	0.0021469349	0.0529211722	0.0010934038
MSE Mish(x)	0.003081422	0.00807074	0.003229788
Train Epoch Time	19.16s	6.18s	17.99s

Table 1: Comparison of LIF-only SNNs against a dendrite-enabled SNN on $\sqrt{x}$ and Mish(x). Test data was created via taking a 1,000 step linear space sample between 0 and a random end value less than 1. The average time to train an epoch is also shown to illustrate computational complexity.

After training, we took the output values of the network against a 1,000 input "test" dataset. This dataset consists of a random sample of 1,000 values, selected from a random linear space with a maximum value of 1.0. In table1, the total error of each network is reported. The 16 LIF Neuron hidden layer network performed worst, as would be expected for a smaller network. The dendritic network and LIF-only SNN networks performed comparably, achieving similar performances. This demonstrates the improved expressivity of SNNs when equipped with dendritic input trees. By simply moving the computation of the LIF neurons into the dendrite, the network is able to achieve accuracy parity with the larger network despite sending only 16 values to the next layer.

5 Conclusion

In this work, we have presented a library that enables simplified hardware-informed dendritic computation in a machine learning enabled software environment. This library models a dendrite component in torch, based on a sub-threshold CMOS transistor capable of 50 pJ for a single compartment that provides two multiplicative operations per compartment. It is thus able to offer the low-power potential of hardware dendrites in traditional machine learning applications. Even with our simplified time-stepped model, dendritic computation allows for far more expressivity with fewer total neurons in a network than an equivalent LIF only network. Our experiments show that the addition of dendrites to a simple SNN network could provide similar performance to the LIF-only SNN network, but with fewer total spikes sent across the network. We hope this tool will facilitate the development of dendrite-enabled hardware-informed deep learning SNN, thus paving the way for the next-generation of energy-efficient deep learning approaches.

Our current work examines the non-linear capabilities of dendritic computation for machine learning applications. Expanding on our simple examples, dendrites can further provide multiplicative behaviors [8], and could potentially be used to replace attention layers in spiking networks [40]. Replacing these layers with dendrites could reduce energy and area on the chip while also increasing expressivity and connectivity of neurons. This would entail designing a neuromorphic accelerator which implements neural components such as dendrites, synapses, soma, axons etc. natively in hardware. Designing hardware such as this requires future co-design of hardware and algorithms, and the creation and design of novel SNN networks.

In addition to hardware support, software support will be critical to drive SNN development and adoption. While there are a plethora of SNN deep learning libraries currently available, there is a lack of hardware-aware dendritic deep learning libraries. We wish to pursue deeper and more complex demonstrations of large-scale convolution and attention-based dendrite SNNs, to fully demonstrate the capabilities of this type of system. Finally, we are working to extend a neuromorphic hardware device performance estimation tool [5], that will provide energy and latency estimates for dendritic SNN models. More work is needed to develop efficient, beyond CMOS hardware devices that can leverage both the computational capabilities and energy efficiency gains that are available in dendrite spiking networks.

Acknowledgments and Disclosure of Funding

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U.S. Government purposes. The DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan.

References

[1] Jyotibdha Acharya, Arindam Basu, Robert Legenstein, Thomas Limbacher, Panayiota Poirazi, and Xundong Wu. Dendritic computing: Branching deeper into machine learning. *Neuroscience*, 489:275–289, 2022. Dendritic contributions to biological and artificial computations.

[2] Gal Ariav, Alon Polsky, and Jackie Schiller. Submillisecond precision of the input-output transformation function mediated by fast sodium dendritic spikes in basal dendrites of ca1 pyramidal neurons. *Journal of Neuroscience*, 23(21):7750–7758, 2003.

[3] John V Arthur and Kwabena Boahen. Recurrently connected silicon neurons with active dendrites for one-shot learning. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 3, pages 1699–1704. IEEE, 2004.

[4] Kwabena Boahen. Dendrocentric learning for synthetic intelligence. *Nature*, 612(7938):43–50, 2022.

[5] James Boyle, Mark Plagge, Suma George Cardwell, Frances S Chance, and Andreas Gerstlauer. Performance and energy simulation of spiking neuromorphic architectures for fast exploration. In *Proceedings of the 2023 International Conference on Neuromorphic Systems*, pages 1–4, 2023.

[6] Suma G Cardwell and Frances S Chance. Dendritic computation for neuromorphic applications. In *Proceedings of the 2023 International Conference on Neuromorphic Systems*, pages 1–5, 2023.

[7] Biswadeep Chakraborty, Xueyuan She, and Saibal Mukhopadhyay. A fully spiking hybrid neural network for energy-efficient object detection. *IEEE Transactions on Image Processing*, pages 9014–9029, 2021.

[8] Frances S Chance and Suma G Cardwell. Shunting inhibition as a neural-inspired mechanism for multiplication in neuromorphic architectures. In *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference*, pages 41–46, 2023.

[9] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*, 2021.

[10] Suma George, Jennifer Hasler, Scott Koziol, Stephen Nease, and Shubha Ramakrishnan. Low power dendritic computation for wordspotting. *Journal of Low Power Electronics and Applications*, 3(2):73–98, 2013.

[11] Mariana-Iuliana Georgescu, Radu Tudor Ionescu, Nicolae-Catalin Ristea, and Nicu Sebe. Non-linear neurons with human-like apical dendrite activations, 2021.

[12] Will Greedy, Heng Wei Zhu, Joseph Pemberton, Jack Mellor, and Rui Ponte Costa. Single-phase deep learning in cortico-cortical networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24213–24225. Curran Associates, Inc., 2022.

[13] Karan Grewal, Jeremy Forest, Benjamin P. Cohen, and Subutai Ahmad. Going beyond the point neuron: Active dendrites and sparse representations for continual learning. *bioRxiv*, 2021.

[14] Michael Häusser and Bartlett Mel. Dendrites: bug or feature? *Current opinion in neurobiology*, 13(3):372–383, 2003.

[15] Chaofei Hong, Mengwen Yuan, Mengxiao Zhang, Xiao Wang, Chegnjun Zhang, Jiaxin Wang, Gang Pan, Zhaohui Wu, and Huajin Tang. Spaic: A spike-based artificial intelligence computing framework. *arXiv preprint arXiv:2207.12750*, 2022.

[16] Junkai Ji, Shangce Gao, Jiujun Cheng, Zheng Tang, and Yuki Todo. An approximate logic neuron model with a dendritic structure. *Neurocomputing*, 173:1775–1783, 2016.

[17] Jakob Kaiser, Sebastian Billaudelle, Eric Müller, Christian Tetzlaff, Johannes Schemmel, and Sebastian Schmitt. Emulating dendritic computing paradigms on analog neuromorphic hardware. *Neuroscience*, 489:290–300, 2022.

[18] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3953–3962, 2021.

[19] Xinyi Li, Jianshi Tang, Qingtian Zhang, Bin Gao, J Joshua Yang, Sen Song, Wei Wu, Wenqiang Zhang, Peng Yao, Ning Deng, et al. Power-efficient neural network with artificial dendrites. *Nature Nanotechnology*, 15(9):776–782, 2020.

[20] Xinyi Li, Yanan Zhong, Hang Chen, Jianshi Tang, Xiaojian Zheng, Wen Sun, Yang Li, Dong Wu, Bin Gao, Xiaolin Hu, et al. A memristors-based dendritic neuron for high-efficiency spatial-temporal information processing. *Advanced Materials*, page 2203684, 2022.

[21] Michael London and Michael Häusser. Dendritic computation. *Annu. Rev. Neurosci.*, 28:503–532, 2005.

[22] Davide Liberato Manna, Alex Vicente-Sola, Paul Kirkland, Trevor Joseph Bihl, and Gaetano Di Caterina. Frameworks for SNNs: a review of data science-oriented software and an expansion of spyketorch. *arXiv preprint arXiv:2302.07624*, 2023.

[23] Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXive preprint: arXive: 1908.08681*, 2020.

[24] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in neuroscience*, page 625, 2019.

[25] Stephen Nease, Suma George, Paul Hasler, Scott Koziol, and Stephen Brink. Modeling and implementation of voltage-mode cmos dendrites on a reconfigurable analog platform. *IEEE transactions on biomedical circuits and systems*, 6(1):76–84, 2011.

[26] Michalis Pagkalos, Spyridon Chavlis, and Panayiota Poirazi. Introducing the dendrify framework for incorporating dendrites to spiking neural networks. *Nature Communications*, 14(1):131, 2023.

[27] Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14:653, 2020.

[28] Christian Pehle and Jens Egholm Pedersen. Norse - A deep learning library for spiking neural networks. `https://github.com/https://github.com/norse/norse`, January 2021. Documentation: https://norse.ai/docs/.

[29] Shubha Ramakrishnan, Richard Wunderlich, Jennifer Hasler, and Suma George. Neuron array with plastic synapses and programmable dendrites. *IEEE transactions on biomedical circuits and systems*, 7(5):631–642, 2013.

[30] Subhrajit Roy, Amitava Banerjee, and Arindam Basu. Liquid state machine with dendritically enhanced readout for low-power, neuromorphic vlsi implementations. *IEEE transactions on biomedical circuits and systems*, 8(5):681–695, 2014.

[31] Subhrajit Roy and Arindam Basu. An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE transactions on neural networks and learning systems*, 28(4):900–910, 2016.

[32] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[33] Johannes Schemmel, Laura Kriener, Paul Müller, and Karlheinz Meier. An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based non-linear dendrites. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2217–2226. IEEE, 2017.

[34] Nelson Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221, 2008.

[35] Marcel Stimberg, Romain Brette, and Dan FM Goodman. Brian 2, an intuitive and efficient neural simulator. *Elife*, 8:e47314, 2019.

[36] Greg Stuart, Nelson Spruston, and Michael Häusser. *Dendrites*. Oxford University Press, 2016.

[37] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[38] Yingxue Wang and Shih-Chii Liu. Active processing of spatio-temporal input patterns in silicon dendrites. *IEEE transactions on biomedical circuits and systems*, 7(3):307–318, 2012.

[39] Xundong Wu, Xiangwen Liu, Wei Li, and Qing Wu. Improved expressivity through dendritic neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[40] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10221–10230, 2021.

[41] Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. Multi-attention recurrent network for human communication comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[42] Tianle Zhou, Shangce Gao, Jiahai Wang, Chaoyi Chu, Yuki Todo, and Zheng Tang. Financial time series prediction using a dendritic neuron model. *Knowledge-Based Systems*, 105:214–224, 2016.