Spiking Deep Neural Networks

Carl Anderson
EE5329
University of Minnesota

September 14, 2022

1 Introduction

The deep learning revolution has been sparked in the past twenty years by the blossoming of Moore's law and the advent of mass data generation and collection. Deep learning relies on enormous data sets and state of the art computing machinery in order to produce extremely accurate models for various machine learning tasks. Spiking neural networks have been suggested as an alternative paradigm of deep learning due to their adjacency to how real brain synapses and neurons function. The human brain is estimated to have over a hundred teraflops of computing ability while being an order of magnitude more energy efficient. New promising neuromorphic (brain-like) hardware is built to be as energy efficient as possible, however it is extremely difficult to map traditional deep learning architectures to neuromorphic hardware, owing to the difference in design methodologies. Spiking neural networks have been proposed to bridge this gap and realize the energy efficiencies of neurmorphic hardware while achieving accuracy on-par with traditional deep learning methods.

At their core, the difference between spiking and traditional rate-based neural networks is the introduction of the time dimension. Rate-based neural networks are time-invariant, they receive an input, perform mathematical operations, and produce a predicted output. Even time-series models such as recurrent neural networks or transformers which account for ordered information still only process each input in a single time-step. Spiking neural networks, however, take an input and propagate that input through the neural network over some set amount of time. Instead of fixed, floating-point 'rates' propagating through the network, each neuron produces stochastic spike trains which transmit information between layers in the number and time between spikes. Spike trains travel through 'synapses' modeled as weights and then to other spiking neurons. These spikes are encoded in a single binary digit format, meaning that all arithmetic is only doing addition, there is no need to perform multiplication because each neuron output is always either zero or one.

2 Spiking Networks

The basic unit in spiking networks is the spiking neuron, as opposed to a perceptron in rate-based architectures. Many different spiking neuron models have been suggested but the most common in nearly all spiking neural network applications is the leaky integrate-and-fire (LIF) neuron. The LIF neuron is named because the membrane voltage accumulates or integrates the input current less the current membrane voltage and once it passes the threshold voltage fires the output synapse [1]. The equation for the membrane potential is given by the following equation:

$$\tau_{RC}\frac{dv_{mem}(t)}{dt} = -v_{mem}(t) + J(t) \tag{1}$$

The LIF neuron model is desirable for many reasons over alternative models. It is much more biologically plausible than integrate-and-fire models, however it is far less computationally intensive compared to more complex models such as the original Hodgekin and Huxley model [2]. Spike trains in the brain are complex signals but in spiking neural networks we can simplify computations by modeling them as sequences of shifted and scaled Dirac delta functions.

There are two primary ways in which machine learning models learn the system they are modelling: supervised and unsupervised. Traditionally, neural networks have been a supervised learning method, utilizing the propagation algorithm to update parameters by feeding lots of labelled inputs through and backpropagating the error term back through the network, updating weights as you go. The first challenge when performing direct supervised training on spiking neural networks is the non-linearity of the spike train signals between layers. The backpropagation algorithm relies on stochastic gradient descent and the differentiability of the cost function with respect to the parameters of the network. However, Dirac delta functions are discontinuous and their derivative does not exist [3]. This issue has been primarily solved by using differentiable alternatives, called 'surrogate gradient descent,' to the Dirac delta function such as converting discrete spikes to continuous probabilities, and these methods while less biologically plausible, are still useful in applications [4, 5]. The second and more critical problem is known as the 'weight transport' problem where symmetric feedback weights must be known for the backporpogation algorithm to work [3]. An early algorithm worked by learning to discriminate the histograms of spike time responses and was proven to be effective for character recognition [6]. Many algorithms for direct spike-based learning have been proposed since such as SpikeProp [7], ReSuMe [8], and SPAN [9]. SpikeProp is a relatively intuitive extension of the standard backpropagation algorithm which attempts to learn a set of spike firing times by backpropagating the spike firing time errors through the network from the output nodes [7]. ReSuMe and SPAN both work on the wWidrow–Hoff update rule from non-spiking network backpropagation, where ReSuMe adapts the rule to work for spiking networks using a form of STDP, and SPAN adapts the network to work for the rule by switching from Dirac delta functions to alpha kernels [8, 9]. Another key component of spiking networks is lateral inhibition, which is when a spiking neuron can prevent adjacent neighbors from spiking for a short time after in order to prevent neurons in the proceeding layer from confusing the signals.

Another proposed method of supervised learning in spiking neural networks is performing learning in a non-spiking domain and then converting weights into the spiking domain for model deployment. Approaches include tailoring a CNN to allow trained parameters to be directly mapped onto SNN parameters after standard CNN training [3]. The first approach for converting CNN weights to a SNN was outlined in a 2015 paper by Cao et. al in which they replaced hyperbolic tangent activation function with the rectified linear unit in order to eliminate negative layer outputs which are costly to represent in the spiking domain, removed biases from the convolutional layers again to prevent negative layer outputs, and replaced max pooling with linear subsampling to reduce model complexity [10].

Oftentimes unsupervised learning can be preferable to supervised learning if possible because labelling data can have a high cost. There have also been a number of unsupervised learning methods proposed for SNN's with the primary motivator being a biologically plausible learning method called spike timing determined plasticity (STDP) [11]. STDP works by increasing weights between neurons when the pre-synaptic neuron fires right before the post-synapse neuron fires, and decreasing weights when the pre-synaptic neuron fires right after the post-synapse neuron fires. This is explained because

when a pre-synaptic neuron fires right before the post-synaptic neuron we say they have high temporal causality, whereas if it fires right after the pre-synaptic neuron fires, they have random or are temporally non-causal [3]. The weight update equations for STDP-type unsupervised learning are given as follows:

$$\Delta w = Ae^{\frac{-|t_{pre}-t_{post}|}{\tau}}, t_{pre} - t_{post} \leq 0, A > 0 \tag{2}$$

$$\Delta w = Ae^{\frac{-|t_{pre}-t_{post}|}{\tau}}, t_{pre} - t_{post} > 0, A < 0 \tag{3}$$

I the timing between the pre-synaptic fire and the post-synaptic neuron fire are positive, the weight is increased and if it is negative, the weight is decreased. This method for unsupervised learning has been shown to be effective in spiking CNN's for image classification, scoring equivalent or better than traditionally trained CNN's in some computer vision tasks [12]. In addition to supervised and unsupervised learning there is a third category of learning which melds the two called semi-supervised learning in which a model is trained using a mix of labelled and unlabelled data. Semi-supervised learning has some significant benefits:less labelled data is needed compared to supervised methods which allows you to reduce the cost with respect to accuracy or increase accuracy with respect to cost, and unlike unsupervised models the outputs will have human-understandable labels because of the inclusion of labelled training data. Semi-supervised learning methods for SNN's have primarily used a combination of STDP unsupervised learning before or after some amount of standard backpropagated supervised learning to attach labels to outputs [3]. One potential downside of STDP, as first introduced in [13], is that weights can be reduced below zero, which as mentioned above can cause issues when mapping to neuromorphic hardware. Though this can be remedied by shifting weights into a purely positive range. Another more recent technique proposed by [14] directly combines surrogate gradient descent and parameter conversion to achieve comparable accuracy to other SNN training methods over far fewer time steps.

3 Hardware

It is impossible to talk about spiking networks without also considering the underlying neuromorphic hardware that they are designed for. This is because depending on the compile your network to hardware and the specific hardware you compile to, your accuracy and energy-efficiency can vary wildly. The basic unit in neuromorphic hardware is called a crossbar array, which is a grid formed by rows of inputs from pre-synaptic neurons and outputs which go to post-synaptic neurons, with the rows and columns connected at each junction by some sort of memory device which stores the weight for that connection between neurons. The crossbar can be thought of as a grid of synapses connecting neurons. There have been many memory devices proposed such as various types of non-volatile RAM [15] and even memristors [16]. Analog crossbars are especially promising due to their significant space and power efficiency over digital arithmetic while having a minimal (<5%) loss in accuracy [17].

In terms of compiling spiking networks onto the neuromorphic hardware, there have been a number of algorithms proposed for different hardware architectures. One such was DFSynthesizer, proposed in [18], which uses a combination of a greedy method of mapping neurons and synapses onto resistive-memory crossbars, then a synchronous dataflow graph (DFG) of the clustered SNN to determine tile execution order, and finally a self-timed execution-based technique to compile the network to hardware at runtime. This method was able to achieve 78% better performance SpiNeMap [19], a method which minimizes inter-tile spikes, and 28% better performance than PYCARL [20], a method which balances

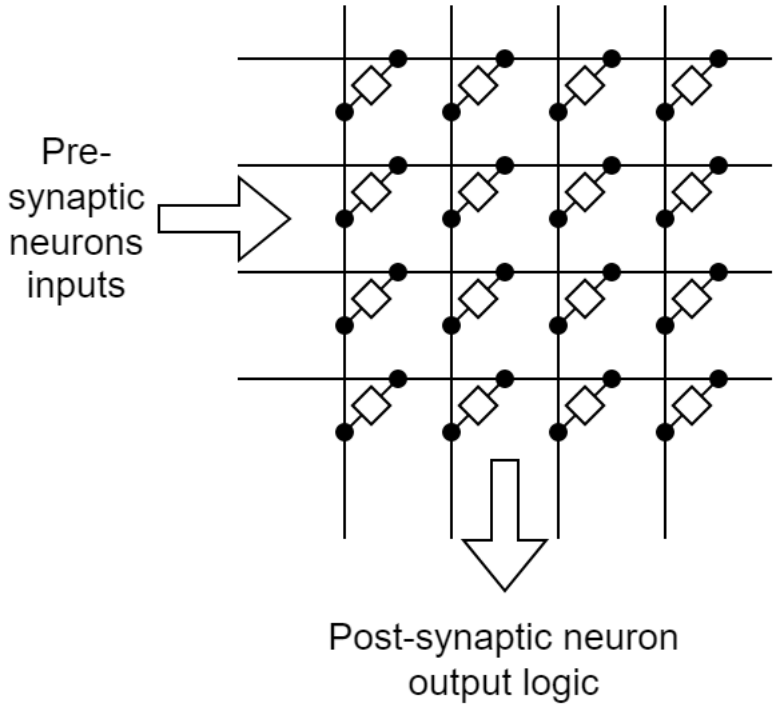

Figure 1: A simple crossbar array architecture with boxes representing memory elements.

number of clusters mapped to each tile.

4 Discussion

Spiking neural networks have been declared the third generation of neural networks, with the first being standard rate-based perceptrons and the second being activation-function gated [21]. Now that it has been proven that a multi-layer perceptron model with non-linear activation functions can approximate any function via the universal approximation theorem [22], the focus on deep learning has shifted towards energy efficiency. Spiking NN's are poised to take over deployed low-energy deployed neural networks due to their efficiency on dedicated neuromorphic hardware. Some applications of lower-power neural networks include internet of things, low-power remote robot vision, and self-driving vehicles. However, spiking networks still have distance to make up compared to standard neural network architectures due to the challenge presented in training these models and compiling the trained network onto hardware. Spiking networks are especially interesting because they sit at the confluence of many advanced fields in both computer science and electrical engineering: deep learning, signal processing, VLSI design, and neuroscience to name a few. Much of the research thus far has considered one section of this pie, the training of the model, the compilation onto hardware, and the design of the neuromorphic hardware itself, but there is still significant room for inter-disciplinary advances to be made when consideration for multiple portions of the complete picture are considered at once.

References

[1] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[3] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural networks*, vol. 111, pp. 47–63, 2019.

[4] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," *Advances in neural information processing systems*, vol. 28, 2015.

[5] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, p. 119, 2020.

[6] D. V. Buonomano and M. Merzenich, "A neural network model of temporal code generation and position-invariant pattern recognition," *Neural computation*, vol. 11, no. 1, pp. 103–116, 1999.

[7] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[8] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[9] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *International journal of neural systems*, vol. 22, no. 04, p. 1250012, 2012.

[10] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[11] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[12] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382–392, 2016.

[13] B. Nessler, M. Pfeiffer, and W. Maass, "Stdp enables spiking neurons to detect hidden causes of their inputs," *Advances in neural information processing systems*, vol. 22, 2009.

[14] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," *arXiv preprint arXiv:2005.01807*, 2020.

[15] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[16] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1864–1878, 2014.

[17] S. Song, A. Balaji, A. Das, N. Kandasamy, and J. Shackleford, "Compiling spiking neural networks to neuromorphic hardware," in *The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, pp. 38–50, 2020.

[18] S. Song, H. Chong, A. Balaji, A. Das, J. Shackleford, and N. Kandasamy, "Dfsynthesizer: Dataflow-based synthesis of spiking neural networks to neuromorphic hardware," *ACM Transactions on Embedded Computing Systems (TECS)*, 2021.

[19] A. Balaji, A. Das, Y. Wu, K. Huynh, F. G. Dell'Anna, G. Indiveri, J. L. Krichmar, N. D. Dutt, S. Schaafsma, and F. Catthoor, "Mapping spiking neural networks to neuromorphic hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 76–86, 2019.

[20] A. Balaji, P. Adiraju, H. J. Kashyap, A. Das, J. L. Krichmar, N. D. Dutt, and F. Catthoor, "Pycarl: A pynn interface for hardware-software co-simulation of spiking neural network," *arXiv preprint arXiv:2003.09696*, 2020.

[21] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[22] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.