EMPIRICAL STUDY ON THE EFFICIENCY OF SPIKING NEURAL NETWORKS WITH AXONAL DELAYS, AND ALGORITHM-HARDWARE BENCHMARKING

Alberto Patiño-Saucedo
CSIC/Universidad de Sevilla
Seville, Spain

Amirreza Yousefzadeh
IMEC-Netherlands
Eindhoven, Netherlands

Guangzhi Tang
IMEC-Netherlands
Eindhoven, Netherlands

Federico Corradi
Eindhoven University of Technology
IMEC-Netherlands

Bernabé Linares-Barranco
CSIC/Universidad de Sevilla
Seville, Spain

Manolis Sifalakis
IMEC-Netherlands
Eindhoven, Netherlands

September 12, 2023

ABSTRACT

The role of axonal synaptic delays in the efficacy and performance of artificial neural networks has been largely unexplored. In step-based analog-valued neural network models (ANNs), the concept is almost absent. In their spiking neuroscience-inspired counterparts, there is hardly a systematic account of their effects on model performance in terms of accuracy and number of synaptic operations. This paper proposes a methodology for accounting for axonal delays in the training loop of deep Spiking Neural Networks (SNNs), intending to efficiently solve machine learning tasks on data with rich temporal dependencies. We then conduct an empirical study of the effects of axonal delays on model performance during inference for the Adding task [1–3], a benchmark for sequential regression, and for the Spiking Heidelberg Digits dataset (SHD) [4], commonly used for evaluating event-driven models. Quantitative results on the SHD show that SNNs incorporating axonal delays instead of explicit recurrent synapses achieve state-of-the-art, over 90% test accuracy while needing less than half trainable synapses. Additionally, we estimate the required memory in terms of total parameters and energy consumption of accomodating such delay-trained models on a modern neuromorphic accelerator [5, 6]. These estimations are based on the number of synaptic operations and the reference GF-22nm FDX CMOS technology. As a result, we demonstrate that a reduced parameterization, which incorporates axonal delays, leads to approximately 90% energy and memory reduction in digital hardware implementations for a similar performance in the aforementioned task.

Keywords Spiking Neural Networks · Synaptic Delays · Axonal Delays · Temporal Signal Analysis · Spiking Heidelberg Digits

1 Introduction

Spiking Neural Networks (SNNs) are models more closely resembling biology than Analog Neural Networks (ANNs) due to their statefulness and binary event-driven encoding of information, which on novel neuromorphic processors, render them highly efficient in temporal processing applications. Lending to more (compact) parameterization SNNs demonstrate competitive performance to deep ANNs (DNNs) [7]; while potentially using fewer MAC operations in digital hardware implementations. Furthermore, the statefulness of SNNs, embodied in the (decaying) membrane potential of neurons, allows them to be mapped to RNNs [8] effectively, even without recurrent synaptic connections. However, for temporal tasks, the best-performing SNN models almost universally include explicit recurrent connections [4, 7, 9–11], which exponentially increases the number of required synaptic weights as a function of the number of neurons, adding a burden to neuromorphic hardware development.

Meanwhile, the role of axonal delays, i.e., the delay for a spike (action potential) to travel from the soma to the axon terminals, which is a critical element of parameterization in biological neural networks, has remained largely unexplored or characterized in the study of the efficacy, model size, and performance of SNNs. This paper attempts

an initial characterization of and effects of synaptic delays on SNN model performance and the impact of accounting for them in neuromorphic processor architectures.

The first contribution of the work in this paper is a simple strategy of training SNN models with axonal delays, which is conformal with back-propagation (BP) frameworks commonly used for SNN/DNN training (BP through-time (BPTT) for DNNs and its extension spatio-temporal BP (STBP) for SNNs). The second contribution regards an assessment and quantification of the effects of synaptic delay parameterization on model performance (accuracy), model complexity (network structure) and model size (number of parameters). The third contribution is a quantification of energy and memory cost of deploying models with synaptic delays on a modern neuromorphic processor, based on two different design strategies.

2   Related Work

Perhaps one of the reasons that delay model training has not been as mainstream in artificial neural network research until now, is the fact that ANN accelerators do not specifically account and optimize for them at the hardware level. By contrast, many digital neuromorphic accelerators provide explicit hardware support for delay structures (dendritic/axonal); either per neuron [12–14], or shared across neurons [5, 15]. This makes delay model training an attractive exploration in relation to compute and power efficiency.

Recurrency in neural networks offers a constrained way of compensating for synaptic delay parameterization, limited to a single-timestep. Despite this limitation, only a handful of works have explored the explicit use of synaptic delays independently of recurrences. One common formalization in the literature of TDNNs [16–18] and delay-aware SNNs [19–22] is to parameterize synapses with an additional learnable delay variable, trainable with back-propagation [20, 23], local Hebbian-like learning rules [24], or annealing algorithms [25]. An alternative approach in TDNNs involves mapping delays in the spatial domain and train them with autoregressive models and so-called temporal convolutions (TCNs) [2, 26–30]. This approach enables structurally simpler models, which are easier/faster to train, but not very compact as their breadth/depth must scale linearly with the number of timesteps needed to capture temporal dependencies. Our approach is akin to this latter strategy but because of the incremental delay quantization-pruning, our models neither narrow the aperture of the temporal window nor make it homogeneous for all neurons (does not lead to deep models).

3   Methods

3.1   Delay Model Description

We use multilayer Leaky Integrate-and-Fire (LIF) Spiking Neural Networks (SNNs). LIF neurons are stateful, and represent a compromise between biological plausibility and computational efficiency for hardware implementation. Their excitation depends on both their time-dependent input I from other neurons and on their internal state, known as the membrane potential u subject to leaky integration with a time constant τ. The equations of the membrane potential update in a discrete-time implementation of a LIF spiking neuron are:

$$u_k = u_{k-1}e^{-\frac{1}{\tau}}(1 - \theta_{k-1}) + I_{k-1} \tag{1}$$

$$\theta_k = \begin{cases} 1 & u_k \geq u_{th} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where θ denotes a function to generate activations or spikes whenever the membrane potential reaches a threshold associated with the neuron, u_{th}.

Multilayer SNNs can be organized as feedforward or recurrent networks. In a recurrent SNN layer, neurons exhibit lateral connectivity, as in Fig. 1(a), and their input is computed by adding the weighted contribution from the N neurons to the previous or pre-synaptic layer and from the M neighboring neurons in their own layer, as shown in the next equation:

$$I_k[recurrent] = \sum_{i=1}^{N} w_i \theta_{i,k} + \sum_{j=1}^{M} w_j \theta_{j,k} \tag{3}$$

To incorporate axonal delays in networks of spiking neurons, we create multiple time-delayed projections or synapses for every pre-synaptic/post-synaptic neuron pair. This way, the activation of a neuron at a given time depends on both its current state and a subset of past activations from neurons in the pre-synaptic layer, with direct projections. The input of a neuron incorporating the proposed model for axonal delays is:

Figure 1: Projections over time in a pre/post-synaptic pair for a) Recurrently connected SNN (R-SNN) and b) Delayed SNN (D-SNN) using receptive fields of stride 2 and depth 4. Weight values are color-consistent.

$$I_k[delays] = \sum_{d \in D} \sum_{i=1}^{N} w_{i,d}\theta_{i,k-d} \tag{4}$$

where $D \in [0,T]$ is the set of delays chosen for a given task. Control over the temporal stride of the delays and the depth of the temporal receptive field is included in the model (see Fig.1(b) for a visualization of the concept). This increases flexibility in the total number of parameters.

3.2 Model Training

We train models that incorporate axonal delays using the following approach, which is compatible with vanilla back-propagation frameworks used to train SNNs and RNNs today (i.e. no special framework extensions). The idea is to express the (temporal) parameterization of delays as a spatial parameterization of synaptic weights, such that delay training is effected by merely optimizing for synaptic weights. We start with a set of parallel synapses per pair of pre and post-synaptic neurons each associated with a delayed output from the pre-synaptic neuron (using a predetermined range of delays and stride). We optimize the model as usual, and prune all delay-synapses that end up with small weights. We then fine-tune the model with only the remaining synapses. We may introduce new synapses to replace the pruned ones, with incrementally higher delay resolution in localized sub-regions of the initial delay range, and repeat the process. As a result different neurons end-up with different fan-in delay inputs. The resulting models are topologically feed-forward, consistently shallower with few parameters than their recurrent-connectivity counterparts, and exhibit state-of-art performance (confirmed in all experiments). Their simpler structure renders them attractive for resource-efficient deployment on neuromorphic accelerators.

We trained SNN models with back-propagation (STBP specifically [31]). This method accounts for the past influence on current neurons' states by unrolling the network in time, and the errors are computed along the reverse paths of the unrolled network. To account for the discontinuity of the membrane potential, we employed a surrogate gradient function [8] with a fast sigmoid function as in [10]. During training, apart from the synaptic weights, we also optimized the membrane's time constants, as in [7]. Finally, we did not consider extra delays for the input at the first layer, as the input layer usually has more neurons and is responsible for a large portion of the synaptic parameters.

3.3 Implementation cost in neuromorphic processors

Neuromorphic hardware architectures implement stateful nodes with scalable event-driven communication, reducing communication and processing costs and, by extension, the required energy. Spiking Neural Networks are some of the most well-suited algorithms for these kinds of processors, and as such, the delay mechanism is supported by most neuromorphic chips. In this paper, we used a simple yet accurate methodology to compare the energy and memory overhead of the delay mechanism. The energy consumption is calculated based on counting the memory accesses (spike packets and neuron states read/write and weights read) and arithmetic operations (accumulation, comparison with threshold, etc.) using a netlist level simulation tool (Cadence JOULES) for an advanced technology node (Global-Foundries 22nm FDX). Memory cost is calculated from the total number of parameters (as shown in Fig.3), the neuron states, and the number of delayed spike packets required to perform inference. We explored two methods commonly used by digital neuromorphic platforms to implement delay: The Ring Buffer [12–14] and the Delay Queue [5, 15].
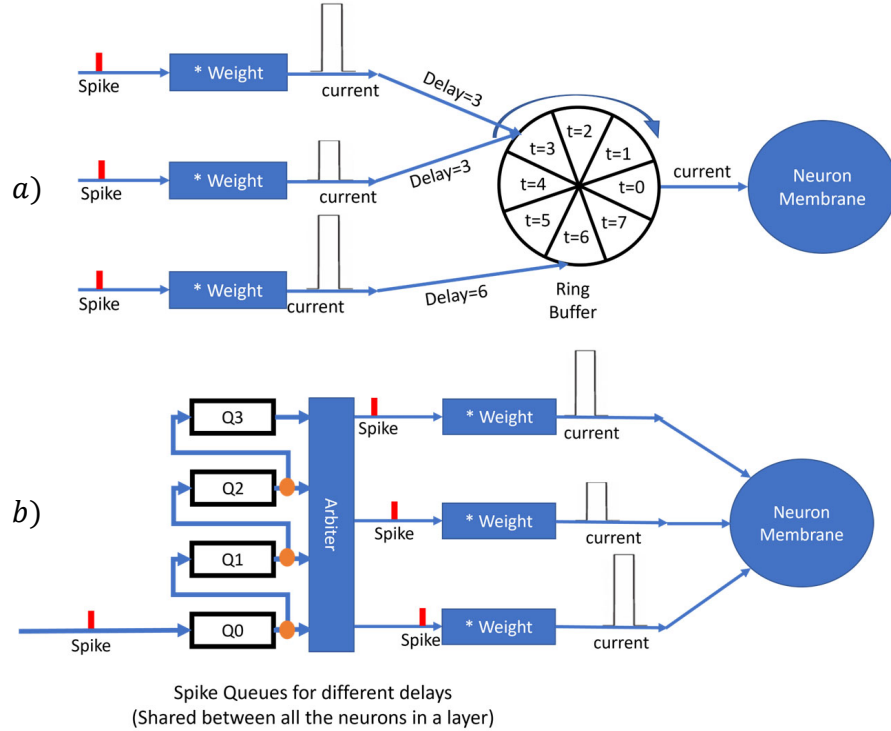
Figure 2: (a) Using a ring buffer per neuron to implement synaptic delays. Unit of delay is the system time-step. (b) Implementation of axon delay by sorting spikes based on the encoded delays in separated delay queues. The queues are shared across neurons in a neuro-synaptic core (not shown in figure).

3.3.1 Ring Buffer

A ring buffer is a special type of circular queue where currents with different delays accumulate in separate elements of the queue. When using the ring buffer, the maximum possible delay in the system will be limited to the size of the buffer, and the set of possible delays is linearly distributed i.e., the temporal stride is constant (see Fig.2(a)). In this method, there is one ring buffer per neuron; therefore, the memory overhead scales with the number of neurons.

The estimated memory overhead for the ring buffer (total sum of the ring buffer sizes) is calculated as "number of postsynaptic neurons with synaptic delay × maximum synaptic delay". The energy overhead is equal to one extra neural accumulation per time step (to accumulate the value of the ring buffer into the membrane potential).

3.3.2 Delay Queue

The axon delay is encoded directly in the spikes in a delay queue. Therefore, each spike packet contains a few bits to indicate the amount of delay. In the destination neuro-synaptic core, instead of having a single queue for all spikes, several queues, each corresponding to a specific delay amount, are implemented. This method is more efficient to implement when spikes activity is sparse. Fig2(b) depicts an implementation of four delay queues. These delay queues are cascaded, are shared by many neurons, and encode an arbitrary amount of delay (does not need to be a linear distribution). In this scheme, unlike the ring buffer, the number of queues is defined based on the number of possible delays and not on the maximum delay amount. However, the size of each queue increases if the queue applies more delay on the spikes (which means the queue needs to keep the spikes for a longer period). Additionally, this method implements the axon delay which is more coarse-grained compared to the dendritic delay implemented by the ring buffer.

To calculate the memory overhead of delay queues, we need to know the number and size of each queue. We assumed that the delay queues are shared between the neurons of a layer. The number of queues is equal to the number of possible delays. Also, since the proposed algorithm assumes that all input spikes are delayed evenly, the total size of all delay queues is equal to the "maximum number of input spikes of the layer in all time-steps × the maximum amount of delay". In this way, there is enough space in the queue to keep the delayed spikes for each time step. We estimate the energy overhead from total number of reads and writes to the delay queues.
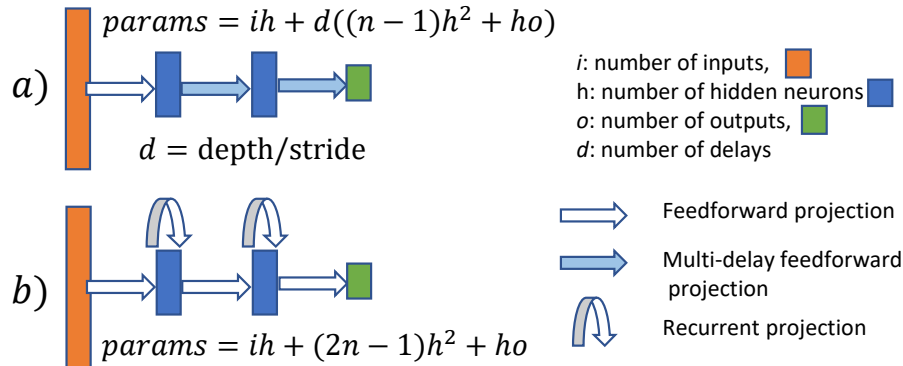
Figure 3: Equations used to calculate the max number of parameters for (a) the delay-based architecture and (b) recurrent SNN proposed here.

4 Results

We report experiments that demonstrate qualitatively the advantages of training SNN models with axonal delays, and quantitatively the benefits from deploying them in digital neuromorphic processors. The first experiment illustrates that models with axonal delays encode more effectively long-term dependencies than networks with recurrent connections. The second experiment reveals that models with synaptic delays achieve state-of-the-art performance, in tasks rich with temporal features, while requiring fewer parameters than recurrent models (similar observations were confirmed on other datasets). This alludes to more compact models, that require less resources for executing on hardware accelerators. A third experiment quantifies this intuition by means with estimates of energy and memory cost, showing a reduction by an order of magnitude, when such models are employed on neuromorphic accelerators, by comparison to *equi-performing* models with recurrent connections. All models were training with the deep learning framework PyTorch on Nvidia GeForce RTX GPU.

4.1 Adding task

The *adding task* is a known benchmark used to evaluate the performance of neural network models on sequence data, such as LSTM [1] or TCN [2, 29]. The input data is a stream of random values chosen uniformly in [0,1], and two randomly selected indexes, one for every half of the sequence. The target, which should be computed at the end of the sequence, is the addition of the two values of the stream at the selected indexes. To use this task to evaluate generic SNNs, we feed the network through two input channels, one for the number stream and the other for the binary-encoded markers, and then compute the Mean Squared Error (MSE) between the target and the membrane potential of a readout neuron with an infinite threshold. Fig. 4 (top) shows that while both a recurrent connectivity and a delay-synapses enable an SNN to remember the indexed numbers and compute the result, the latter however exhibits a more "sensible" or interpretable evolution towards the answer. The bottom of the figure on the other hand, reveals that models with synaptic delays converge typically much faster than traditional ones with recurrent connectivity.

4.2 SHD task

Fig. 5 shows for different models, a comparison of the accuracy on the SHD dataset [4] as a function respectively of the number of model parameters and the number of spikes generated by the model at inference (the number of parameters is a proxy metric for the model size/complexity, and spikes is a proxy metric of energy consumption on any hardware accelerator). The comparison includes various models generated with our method while bounding the max numbers of delay synapses per neuron pair retained after pruning. No pruning refers to retaining all delay synapses. The comparison includes as baseline two recurrent SoA models from the literature [7] that use the adaptive LIF (ALIF) and LIF neuron models. The observation is that with the herein proposed training method we can generate models, which are exceptionally compact, energy-efficient, and yet achieve SoA accuracy. These results are further quantified and distilled in Table 1, where a comparison is made with different feed-forward and recurrent SNN architectures found in the literature for the same dataset.

4.3 Energy estimations of hardware implementation

Table 2 reports the proposed algorithm's estimated energy consumption and memory footprint for both of the commonplace implementations of delay synapses in existing neuromorphic processors discussed in section 3.3. The main take-away observation is that the energy and memory overhead from utilizing synaptic delay hardware structures is substantially off-set by the far more compact, with sparser activity, synaptic delay models. The energy estimations are provided only for comparison purposes and extracted from simulations of digital circuits (SRAM memory accesses
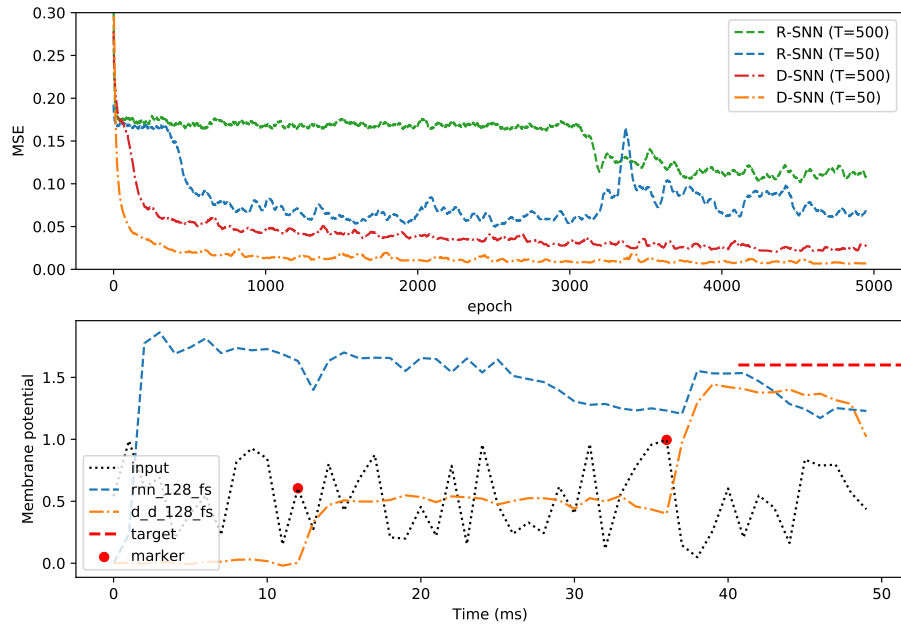
Figure 4: Top: An example of the Adding Task for a sequence length T=50, solved by an R-SNNN (orange) and a D-SNN (green). Notice how the D-SNN "remembers" both values relevant to the task in a more natural way. Bottom: MSE per training epoch for R-SNN (with recurrent synapses) and D-SNN (with delay synapses) in the Adding task, for two sequence lengths: T=50 and T=500. D-SNNs converge faster and to a smaller error!
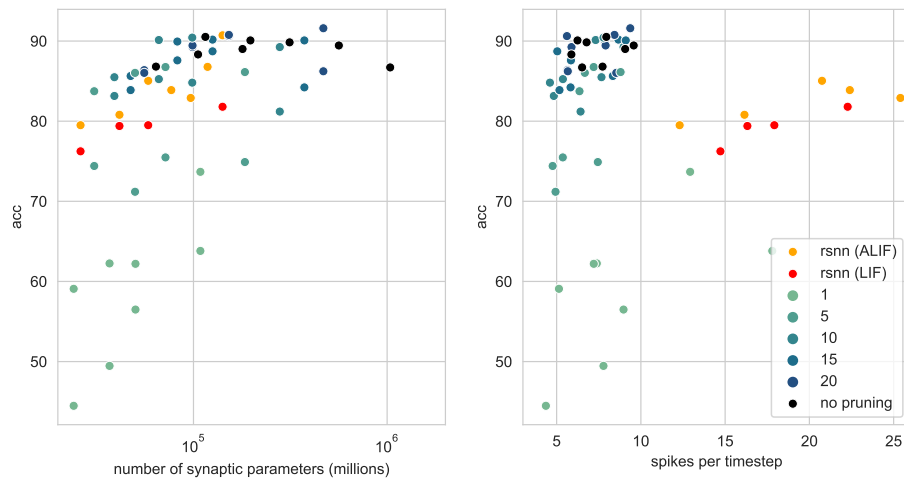

Figure 5: Effect of synaptic delays on performance (SHD task). Left: num of parameters vs accuracy. Right: num of spikes vs accuracy. Red and orange points are recurrently connected SNNs. Colors ranging from green to black are SNNs with axonal delays and different pruning configurations.

Table 1: Comparing accuracy and number of parameters for SHD.

Paper	Neuron Type	Architecture*	T	Params.	Acc.
Eshraghian, 2022	LIF[a]	3000r	100	11160000	83.2
Eshraghian, 2022	LIF[a]	3000	100	2160000	66.3
Bauer, 2022	SRM	100+1281+1281+201	250	2100562	78.1
Zenke, 2022	LIF	1024r	2000	1785856	83.2
Fang, 2021	SRM	400+400	2000	448000	85.7
Yu, 2022	LIF[b]	400+400	1000	448000	87.0
Zenke, 2021	LIF	256r+256r	500	380928	82.0
Yin, 2020	LIF[c]	256r	250	249856	81.7
Yin, 2021	LIF[c]	128r+128r	250	141312	**90.7**
Zenke, 2022	LIF	128r	2000	108544	71.4
Perez, 2021	LIF	128r	2000	108544	82.7
Ours (1)	**LIF**	64d+64d	250	98560	**90.4**
Ours (2)	**LIF**	48d+48d	250	**66240**	90.1

* Conventions: r: with lateral recurrency, d: with delay synapses.
[a] Binarized. [b] MAP-SNN. [c] Adaptive threshold.

Table 2: Energy and memory estimations for the proposed network, compared to an RSNN for similar accuracy.

Measurement	R1	R2	D1	D2
neurons per hidden layer	128	48	8	8
number of delays	1	1	10	5
avg spk/timestep, layer1	8.678	6.725	1.894	1.686
avg spk/timestep, layer 2	4.582	3.456	1.772	2.539
max spk/timestep, layer 1	-	-	7	7
max spk/timestep, layer 2	-	-	7	8
test set accuracy	81.020	80.200	82.170	80.510
Neurosynaptic cost estimation				
energy (uJ)	20.213	7.390	2.304	1.745
memory (param. count)	141588	41684	7876	6756
Delay queue estimations				
energy overhead (uJ)*	-	-	0.059	0.030
mem. overhead (words)	-	-	1890	1800
energy saving factor	1	2.735	8.554	11.384
memory saving factor	1	3.397	14.498	16.548
Ring buffer estimations				
energy overhead (uJ)	-	-	0.085	0.085
mem. overhead (words)	-	-	3780	3360
energy saving factor	1	2.735	8.463	11.046
memory saving factor	1	3.397	12.147	13.996

*The energy overhead is calculated per inference.
All networks evaluated for T=250. Columns:
R1: (Recurrent) LIF 128r+128r.
R2: (Recurrent) ALIF 48r+48r.
D1: (Delays) LIF 8d+8d, depth=150, stride=15.
D2: (Delays) LIF 8d+8d, depth=150, stride=30.

and arithmetic operations in float 16b data type). For memory overhead, we assumed that all parameters, neuron states, and spike packets use the same data types and only report the total number of memory words. Simulations are for CMOS digital technology node GF-22nm FDX, through Cadence software tools.

5   Conclusion

We introduced a method for training SNN models with synaptic delays, and we report benefits of deploying such models in neuromorphic accelerators. The important observation from the resulting trained models is that even a small set of synaptic delays together with trainable time constants, supersede the need for complex lateral connectivity, reduce the number of layers and total number of parameters needed for good performance. This also reduces the memory footprint of these models in neuromorphic accelerators (compared to commonplace RNNs). Future work will focus on *hardware-aware* training of synaptic delay models for compact mappings on neuromorphic accelerators.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[2] S. Bai, J. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 03 2018.

[3] A. Kag and V. Saligrama, "Training recurrent neural networks via forward propagation through time," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5189–5200.

[4] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2022.

[5] A. Yousefzadeh, G.-J. van Schaik, M. Tahghighi, P. Detterer, S. Traferro, M. Hijdra, J. Stuijt, F. Corradi, M. Sifalakis, and M. Konijnenburg, "Seneca: Scalable energy-efficient neuromorphic computer architecture," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 371–374.

[6] G. Tang, A. Safa, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G.-J. van Schaik, and A. Yousefzadeh, "Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[7] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[8] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[9] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," in *International Conference on Neuromorphic Systems 2020*, 2020, pp. 1–8.

[10] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural computation*, vol. 33, no. 4, pp. 899–925, 2021.

[11] N. Perez-Nieves, V. C. Leung, P. L. Dragotti, and D. F. Goodman, "Neural heterogeneity promotes robust learning," *Nature communications*, vol. 12, no. 1, pp. 1–9, 2021.

[12] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[13] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, 2018.

[14] A. Morrison, C. Mehring, T. Geisel, A. Aertsen, and M. Diesmann, "Advancing the boundaries of high-connectivity network simulation with distributed computing," *Neural computation*, vol. 17, no. 8, 2005.

[15] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[16] S. Day and M. Davenport, "Continuous-time temporal back-propagation with adaptable time delays," *IEEE Transactions on Neural Networks*, vol. 4, no. 2, pp. 348–354, 1993.

[17] R. Duro and J. Reyes, "Discrete-time backpropagation for training synaptic delay-based artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 779–789, 1999.

[18] J. Santos and P. Fernandez, "Evolved synaptic delay based neural controllers for walking patterns in hexapod robotic structures," *Natural Computing*, vol. 16, 06 2017.

[19] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 1419–1428.

[20] X. Wang, X. Lin, and X. Dang, "A delay learning algorithm based on spike train kernels for spiking neurons," *Frontiers in Neuroscience*, vol. 13, 2019.

[21] B. Schrauwen and J. Van Campenhout, "Extending spikeprop," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 1, 2004, pp. 471–475.

[22] S. B. Shrestha and Q. Song, "Adaptive delay learning in spikeprop based on delay convergence analysis," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 277–284.

[23] R. Boné and H. Cardot, "Time delay learning by gradient descent in recurrent neural networks," in *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ser. ICANN'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 175–180.

[24] M. Zhang, J. Wu, A. Belatreche, Z. Pan, X. Xie, Y. Chua, G. Li, H. Qu, and H. Li, "Supervised learning in spiking neural networks with synaptic delay-weight plasticity," *Neurocomputing*, vol. 409, pp. 103–118, 2020.

[25] B. Cohen, D. Saad, and E. Marom, "Efficient training of recurrent neural network with time delays," *Neural Networks*, vol. 10, no. 1, 1997.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, 1989.

[27] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[28] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1242–1250.

[29] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2017.

[30] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech 2015*, 2015, pp. 3214–3218.

[31] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, 2018.

*
