Co-optimized training of models with synaptic delays for digital neuromorphic accelerators

Alberto Patiño-Saucedo[a], Roy Meijer[b], Paul Detteter[c], Amirreza Yousefzadeh[c, d] ,
Laura Garrido-Regife[a], Bernabé Linares-Barranco[a], and Manolis Sifalakis[c]
[a] Instituto de Microelectrónica de Sevilla, CSIC/Universidad de Sevilla, Seville, Spain
[b] Eindhoven University of Technology, Dept. Electrical Engineering, Eindhoven, Netherlands
[c] IMEC-Netherlands, Eindhoven, Netherlands [d] University of Twente, Twente, Netherlands
Corresponding author email: manolis.sifalakis@imec.nl

2024 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-3099-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCAS58744.2024.10558209

***Abstract*—Configurable delays are a basic feature in many neuromorphic neural network hardware accelerators. However, they have been rarely used in model implementations, despite their promising impact on performance and efficiency in tasks that exhibit complex dynamics, as it has been unclear how to optimize them. In this work, we propose a framework to train and deploy in digital neuromorphic hardware highly performing spiking neural networks (SNNs) where apart from the synaptic weights, the delays are also co-optimized. We consider synaptic (i.e. per-synapse) delays and evaluate them in two neuromorphic digital hardware platforms: Intel's Loihi and Imec's Seneca. Leveraging spike-based back-propagation-through-time, the training process accounts for both platform constraints, such as synaptic weight precision and the total number of parameters per core, as a function of the network size. In addition, a delay pruning technique is used to reduce memory footprint with a low cost in performance. The evaluated benchmark involves several models for solving the SHD (Spiking Heidelberg Digits) classification task, where minimal accuracy degradation during the transition from software to hardware is demonstrated. To our knowledge, this is the first work show-casing how to train and deploy hardware-aware models parameterized with synaptic delays, on multicore neuromorphic hardware accelerators.**

***Index Terms*—Spiking Neural Networks, Synaptic Delays, Temporal Signal Analysis, Spiking Heidelberg Digits**

I. Introduction

Axonal and dendritic synaptic delays are known to be subject to learning and play a (time-warping) role in the information propagation in biological neural networks [1]. Bearing fidelity to neuroscience inspiration therefore many neuromorphic neural network accelerators, offer primitives and constructs to facilitate synaptic delays [2]–[6].

However, up until recently parameterization of neural networks models with synaptic delays had been a largely unexplored territory (for SNNs and ANNs alike), with only a small corpus of literature historically dedicated in the topic [7]–[15].

Nevertheless recently, a few alternative approaches have started emerging for training such *delay models* using back-propagation [16]–[20], offering a fresh coverage of the topic with promising results. A shared observation that seems to be confirmed in all of these recent works is that models parameterized with synaptic delays achieve competitive, and often superior, performance than other models (both spiking or non-spiking), with on-average smaller model sizes. In addition in [16] it was also observed that such models additionally tend to have sparser activity, which makes them more memory and energy efficient, despite the memory overhead in neuromorphic accelerators for implementing the delay structures.

Motivated from the study in [16], which defended the energy and memory efficiency of delay models executing on neuromorpchic accelerators based on estimations, in this paper we conduct actual experiments with two digital hardware platforms (Intel's Loihi [5] and Imec's Seneca [3]), in order to assess various aspects of the efficiency of delay models. One key question regards the fidelity of delay models trained offline and deployed for inference on neural accelerators. A second question regards the actual energy/power, latency, memory efficiency of deploying delay models on hardware. The main contributions of this paper are

- a hardware-aware training framework generating (synaptic) delay models suited for hardware acceleration
- a first of its kind benchmark of (synaptic) delay models executed on neuromorphic hardware accelerators
- a validation of the credibility of the methodology presented in [21] for algorithm-hardware benchmarking

II. Related Work

A canonical formalization of delays, often seen in the literature has been to parameterize synapses with an additional learnable delay variable, which can be learned with back-propagation [10]–[12], [19], [22], local Hebbian-like learning rules [17], annealing algorithms [7], etc. An alternative approach featured in TDNNs [8], [14], [15], [23], [24], and recently for SNNs [18] involves mapping delays in the spatial domain and train them with autoregressive models as temporal convolution kernels (of fixed receptive fields).

Digital neuromorphic neural accelerator platforms have typically implemented support structures for synaptic delays using one of two methods: *Ring Buffer* [4], [5], [25] or *Shared Delay Queue* [2], [3]. A ring buffer is a special type of circular queue where currents with different delays accumulate in separate slots 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. In this method, there is one ring buffer per neuron; therefore, the memory overhead scales with the number of neurons. By contrast a delay queue, is a shared structure across neurons lending to more area/memory efficiency when spike activity is sparse. The axon delay is encoded directly in each spike packet containing a few extra bits to indicate the amount of delay. The destination neuro-synaptic core has several queues, each corresponding to a specific delay amount, connected in a cascade. In this scheme, the number of queues scales with the number of possible delays and not the maximum delay. The size of each queue increases if the queue applies more delay on the spikes.

In [21] a useful hardware-algorithm benchmarking methodology was introduced, which was adopted in [16] for producing estimates of the memory and energy cost of deploying

Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on September 24,2024 at 14:06:08 UTC from IEEE Xplore. Restrictions apply.

synaptic delay models on a neuromorpchic accelerator. In this work we follow this methodology to generate similar reference estimates for the networks that we deployed and constrast them with the actual measurements that we performed.

III. Methods

A. Delay Model Description

In this work, we use a multilayer Spiking Neural Network (SNN) based on Leaky Integrate-and-Fire (LIF) neurons. Their dynamic depends on their internal state, known as the membrane potential, u, and their time-dependent input, I. Also, it considers leaky integration that depends on a time constant, τ. This model has a nature that permits computational efficiency for hardware implementation and, at the same time, reproduces biological processes. In a discrete-time realization of a LIF neuron, the membrane potential updates as follows:

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

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

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

To incorporate synaptic delays in a conventional feed-forward SNN, we create multiple time-delayed projections or synapses for every pre-synaptic/post-synaptic neuron pair. Consequently, the firing of a neuron at a specific time depends on both its current state and a subset of past activations from neurons in the pre-synaptic layer, connected through direct projections. The input of a neuron implementing the suggested framework for synaptic delays is:

$$I_k = \sum_{d \in D} \sum_{i=1}^{N} w_{i,k}^{(d)} \theta_i^{(d)} \tag{3}$$

where $D = \{0, s, 2s, ...d_{max}\}$ is the set of delays chosen for a given task, which is defined with the maximum delay, d_{max}, and the stride, s. The set of weights $\{w_{i,j}\}$ in a classic SNN is replaced by $\{w_{i,j}^{(d)}\}$, since the group of weights can take different values for each delay (see Fig.1(a)). This increases flexibility in the model by allowing to control the total number of parameters.

B. Training Framework

We train models that incorporate synaptic delays using the following strategy, which is compatible with vanilla training frameworks used to train SNNs and RNNs today (i.e. no special framework extensions), such as back-propagation in our case.

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, see Fig.1(a)). 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 (see Fig.1(b)). 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 fewer parameters than their recurrent-connectivity counterparts, and exhibit state-of-art performance (observed in in [16] and confirmed in other related literature [18], [19], [26]). Their simpler structure renders them attractive for resource-efficient deployment on neuromorphic accelerators.

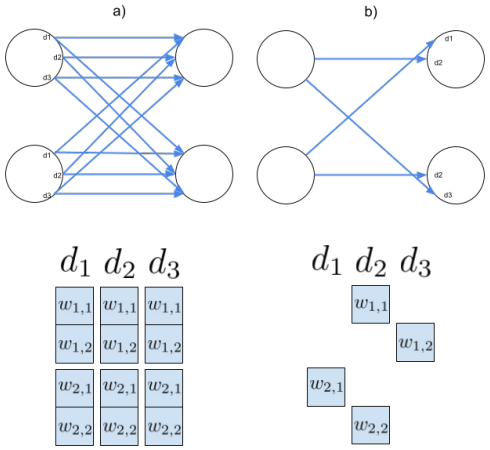

Fig. 1. Weight-delay representation before (a) and after (b) prune delay-synapses.

Note that this training strategy for the delays allows us a-priori to take into account hardware constraints such as the maximum delay supported in the platform (e.g. 60 timesteps in Loihi) as well as the delay memory constraints (stride and pruning level may be amortized so that the resulting spike activity does not exceed the available memory).

Another appealing consequence of the simplicity of this strategy is that it is also compatible with on-device local learning rules such as STDP [27], [28], as it reduces the goal of learning delays to one of adapting weights.

We train our SNN synaptic delay capable models with spatio-temporal back-propagation (STBP) [29], a back-propagation through time (BPTT) variant for SNNs. To account for the discontinuity of the membrane potential, we employed a surrogate gradient function [30] with a fast sigmoid function as in [31].

C. Network Models and Dataset: SHD

Using the delay model and training framework that was introduced in the previous sections, we trained three network models for the Spiking Heidelberg Digits (SHD) benchmark [32]: 700-48-48-20, 700-32-32-20, and 700-24-24-20. In each network, synaptic delays are employed between the hidden layers and between the second hidden and output layers. We did not consider synaptic delays from input to the first hidden layer, as the input layer usually has many neurons and is responsible for a large portion of the synaptic parameters. Table I details the trained network topologies, training configurations, initial delay configuration per hidden layer, as well as the final number of delay levels retained after the iterative synapse pruning, and the reference accuracy achieved for each model on the test-set. The initial delay configurations are provided as max delay level and stride, which for example (60, 2) would

Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on September 24,2024 at 14:06:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I
THE MODELS TRAINED WITH SYNAPTIC DELAYS FOR THE EXPERIMENTS

Parameters	Model 1	Model 2	Model 3
Topology	700-48-48-20	700-32-32-20	700-24-24-20
Num timesteps	64	64	64
Max delay, stride	60,2	60,2	60,2
# Delays after prune	15	15	15
# Parameters	82.6K	47.4K	32.6K
Ref. Accuracy	87%	82%	83%

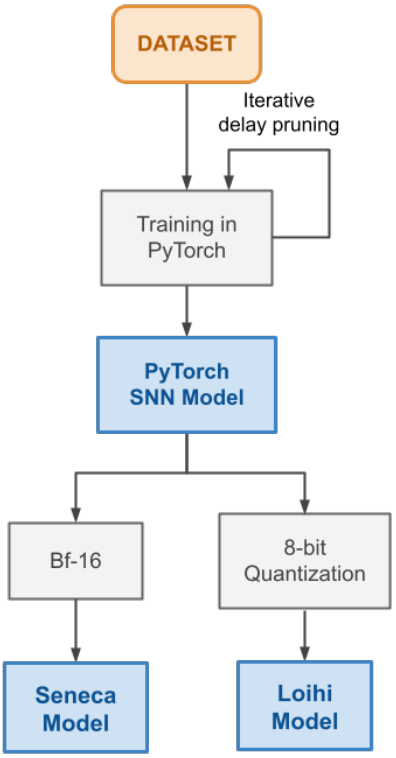

Fig. 2. Delay model training pipeline.

imply that we utilize synapses with delay levels {0,2,4 .. 58}. Fig. 2 illustrates the steps in the training pipeline.

D. Mapping to Seneca and Loihi

We used PyTorch to train the SNN models with synaptic delays. After training, we perform hardware-aware fine-tuning with weight quantization and type conversion so as to produce models that can execute on the neuromorphic accelerators that we used. In the case of Imec's Seneca, we quantized the models to 16-bit Brain-float, and in the case of Intel's Loihi to 8-bit integer (see Fig.2). On Seneca, we map the networks to 3 cores (one layer per core, and each core is served by one shared delay queue). In Loihi, we could also map the network to 3 cores (one for the first layer, one for 37/48 neurons of the second layer, and one for the remaining neurons of the second layer plus the third layer).

Loihi implements delays with a structure similar to ring-buffers, while Seneca uses a variation of shared delay queues.

IV. RESULTS

For the tests on Loihi, we had available a Kapoho Bay USB stick with dual Loihi chip 128 cores each, and remote access to a Nahuku board (32 chips) via the Intel Neuromorphic Research Community. Imec's Seneca is not taped out in ASIC yet, but we are able to run circuit-level simulations for CMOS digital technology node GF-22nm FDX using Cadence Xcelium in 500MHz.

TABLE II
FIDELITY OF LOIHI/SENECA RUN MODELS AGAINST THE PYTORCH TRAINED "MOTHER-MODEL"

Model	Pytorch	Loihi	Seneca
Model accuracy			
700-48-48-20	87%	87%	86%
700-32-32-20	82%	83%	81%
700-24-24-20	83%	81%	82%
Prediction consistency against pytorch			
700-48-48-20	100%	97%	93%
700-32-32-20	100%	95%	97%
700-24-24-20	100%	93%	98%
Avg spike count per inference per layer			
	h1 h2	h1 h2	h1 h2
700-48-48-20	545.32 534.55	524.39 531.14	544.71 534.25
700-32-32-20	400.99 384.61	398.33 365.15	401.18 384.85
700-24-24-20	273.86 311.19	272.48 300.75	271.99 307.9

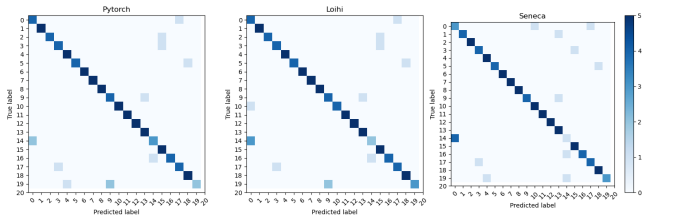

Fig. 3. Confusion matrices for PyTorch trained "mother-model" and Loihi/Seneca run models for the 700-48-48-20 network.

A. Fidelity of delay models on neuromorphic hardware

The first aspect that we were concerned with verifying, was the fidelity in terms of model performance, between the PyTorch-trained model ("mother model") and the fine-tuned models for the two neuromorphic accelerators. Table II summarises the rather positive observations. Both quantized models running on Loihi and Seneca were very close in behavior and final task performance to the "mother model". There is about 1% deviation in task accuracy between inference in PyTorch and inference in the neuromorphic accelerators. Very importantly, the predictions made by the hardware models were very consistent with those of the "mother model". This was well above 90% consistency across all classes, for all models. A more detailed breakdown per class for the 700-48-48-20 network is shown in Fig 3.

Last and even more important is the fact that the spike traces and membrane evolution traces inside the network are also very consistent between "mother model" and the hardware executing models. This can be verified both in the numbers reported Table II for the entire test-set and the three models, and also visually in Fig 4 and 5 for one example data point. It means that the software model can provide us with very reliable information about the anticipated energy per inference and memory use on the neuromorphic platform without even running the hardware model (since these traces are a proxy for the number of synaptic operations).

B. Power, Energy, Latency, Memory measurements

For power measurements on Loihi, we used Intel's cloud Loihi server farm, as it is currently the only way to perform power measurements of neurosynaptic cores. We repeated all the tests one hundred times and averaged the results to account

Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on September 24,2024 at 14:06:08 UTC from IEEE Xplore. Restrictions apply.

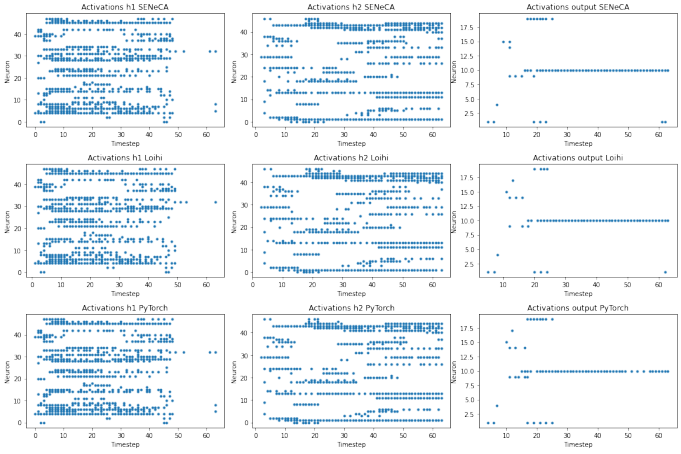

Fig. 4. Comparison of spike traces between pytorch "mother-model" and the one executed on Seneca and Loihi for SHD testset sample X.

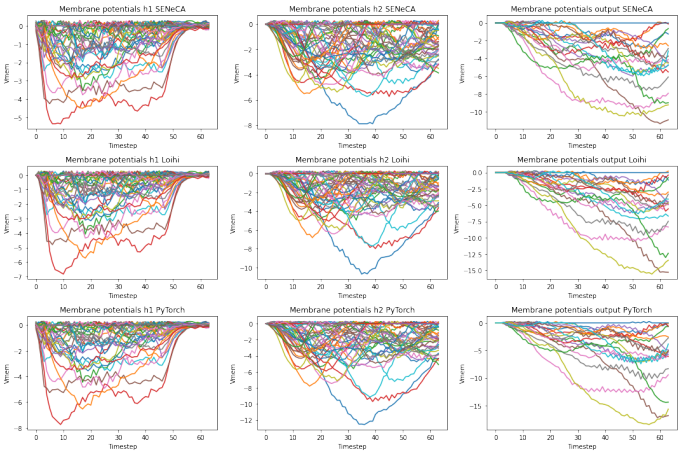

Fig. 5. Comparison of Vmem trace between pytorch "mother-model" and the one executed on Seneca and Loihi for SHD testset sample X.

for variance in power measurements. On the other hand, for Seneca, we used the Cadence JOULE tool, which is accurate within 15% of signoff power. This allowed us to obtain very fine-grained power/latency measurements.

Our measurement results for the 700-48-48-20 network on Loihi and Seneca are presented in Table III. The network is mapped on three neurosynaptic cores in both cases. The power consumption reported in the table includes only the total power of the neurosynaptic cores, excluding the I/O and peripherals.

Regarding Seneca, we found that most of the energy consumption (55uJ out of 69.6uJ) is due to pre/post-processing events by the RISC-V processor, leaving only 14.6uJ for neural processing operations. Similarly, out of the 3.3Mb

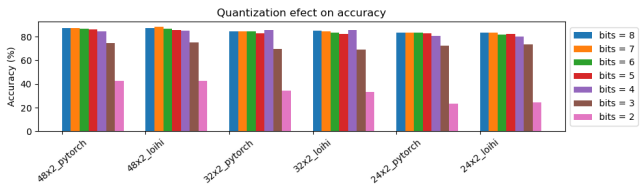

Fig. 6. Effect of weight quantization on final accuracy. All test-set (2264 samples) are considered.

TABLE III
HARDWARE MEASUREMENTS FOR EXECUTING INFERENCE WITH MODEL 700-48-48-20 (*: DOES NOT INCLUDE THE LAKERMONT)

HW measurement	Loihi	Seneca
model: 700-48-48-20		
Power (static+dynamic)	4.22mW (*)	16.30mW
Energy per inference	38.4uJ	69.6uJ
Latency per inference	9.1ms	4.27ms
Memory consumption	4.2Mb	3.3Mb
Synaptic Delay energy	NA	0.98uJ

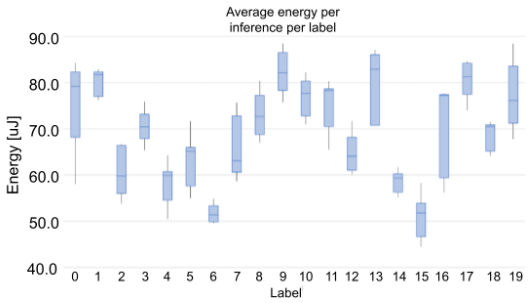

Fig. 7. Energy consumption per inference by class on Seneca.

memory consumption, RISC-V instructions take up 0.5Mb, while neural processing operations use up 2.6Mb. It's worth noting that the actual energy and memory consumption of neural processing operations (14.6uJ/2.6Mb) are very close to the estimates generated using the methodology introduced in [21] (14.48uJ/2.2Mb). Although there is significant overhead, there are several optimization methods that can be applied to reduce the RISC-V overhead, which are out of the scope of this paper.

Fig 7, provides a more detailed breakdown of the energy boxplots for the per class data points in the testset for Seneca.

C. Final accuracy and effect of reduced bit precision

We report the accuracy on the entire test set (2264 samples) and the effect of quantizing with different bit precisions for the three base models implemented on Loihi. As observed in Fig. 6 and consistent with the fidelity results, the degradation in accuracy of the mapping form Pytorch to Loihi is negligible. Interestingly, the effect of reducing the bit precision is, at most less than 3% percent for up to 4 bits, leading (potentially) to a further reduction of the memory footprint with little cost. The maximum accuracy obtained in Loihi for the 700-48-48-20 model is 89.04%

V. CONCLUSION

In this paper, a framework for training and deploying highly performing spiking neural networks (SNNs) with synaptic delays on digital neuromorphic hardware is proposed. The framework co-optimizes both synaptic weights and delays and evaluates them in two neuromorphic digital hardware platforms: Intel's Loihi and Imec's Seneca. The training process leverages spike-based back-propagation-through-time and accounts for platform constraints. A delay pruning technique is used to reduce memory footprint with low cost in performance. The evaluated benchmark involves several models for solving the SHD classification task, where minimal accuracy degradation during the transition from software to hardware is demonstrated. This work demonstrates the first successful application of hardware-aware models parameterized with synaptic delays on multi-core neuromorphic hardware accelerators.

Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on September 24,2024 at 14:06:08 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] C. Stoelzel, Y. Bereshpolova, J.-M. Alonso, and H. Swadlow, "Axonal conduction delays, brain state, and corticogeniculate communication," *The Journal of Neuroscience*, vol. 37, pp. 0444–17, 05 2017.

[2] 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.

[3] G. Tang, K. Vadivel, Y. Xu, R. Bilgic, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G.-J. van Schaik, and A. Yousefzadeh, "Seneca: building a fully digital neuromorphic processor, design trade-offs and challenges," *Frontiers in Neuroscience*, vol. 17, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2023.1187252

[4] 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.

[5] 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.

[6] J. A. Starzyk, Maciura, and A. Horzyk, "Associative memories with synaptic delays," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 331–344, 2020.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] A. Patiño-Saucedo, A. Yousefzadeh, G. Tang, F. Corradi, B. Linares-Barranco, and M. Sifalakis, "Empirical study on the efficiency of spiking neural networks with axonal delays, and algorithm-hardware benchmarking," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[17] 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.

[18] I. Hammouamri, I. Khalfaoui-Hassani, and T. Masquelier, "Learning delays in spiking neural networks using dilated convolutions with learnable spacings," *arXiv preprint arXiv:2306.17670*, 2023.

[19] 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.

[20] P. Sun, E. Eqlimi, Y. Chua, P. Devos, and D. Botteldooren, "Adaptive axonal delays in feedforward spiking neural networks for accurate spoken word recognition," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] P. Sun, L. Zhu, and D. Botteldooren, "Axonal delay as a short-term memory for feed forward deep spiking neural networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8932–8936.

[27] B. Linares-Barranco, T. Serrano-Gotarredona, L. Camuñas-Mesa, J. Perez-Carrasco, C. Zamarreño-Ramos, and T. Masquelier, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Frontiers in Neuroscience*, vol. 5, 2011.

[28] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, 2015.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

*

*This work was funded by EU Horizon grant 101070679 "NimbeAI" and EU H2020 grant 871501 "Memscales"

Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on September 24,2024 at 14:06:08 UTC from IEEE Xplore. Restrictions apply.