Efficient GPU training of LSNNs using eProp

James C Knight
J.C.Knight@sussex.ac.uk
University of Sussex
School of Engineering and Informatics
Brighton, United Kingdom

Thomas Nowotny
T.Nowotny@sussex.ac.uk
University of Sussex
School of Engineering and Informatics
Brighton, United Kingdom

ABSTRACT

Taking inspiration from machine learning libraries – where techniques such as parallel batch training minimise latency and maximise GPU occupancy – as well as our previous research on efficiently simulating Spiking Neural Networks (SNNs) on GPUs for computational neuroscience, we have extended our GeNN SNN simulator to enable spiske-based machine learning research on general purpose hardware. We demonstrate that SNN classifiers implemented using GeNN and trained using the eProp learning rule can provide comparable performance to those trained using Back Propagation Through Time and show that the latency and energy usage of our SNN classifiers is up to 7× lower than an LSTM running on the same GPU hardware.

CCS CONCEPTS

• **Computing methodologies** → *Bio-inspired approaches*; *Supervised learning*; *Vector / streaming algorithms.*

KEYWORDS

spiking neural networks, efficient simulation, GPU

ACM Reference Format:
James C Knight and Thomas Nowotny. 2021. Efficient GPU training of LSNNs using eProp. In *NICE '22: Proceedings of the Neuro-inspired Computational Elements Workshop.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, several new techniques for directly training spiking neural networks (SNNs) using gradient-based approaches have been developed. The discontinuity of spiking neuron's membrane potentials when they emit a spike is problematic when deriving gradients. One approach to work around this problem is to replace the derivative of the membrane potential with a 'surrogate gradient' [3, 5, 27], allowing SNNs to be trained with the same algorithms used to train rate-based Recurrent Neural Network (RNNs) such as Back Propagation Through Time (BPTT). While BPTT is computationally efficient, because it requires gradients to be stored during the forward pass in order for them to be applied during a backward pass, it has a memory requirement which grows with time preventing it from being applied to long input sequences or used online. RTRL [21] is an alternative 'forward mode' algorithm for training RNNs but, in its general form, it is too computationally expensive to be practical. However, if the gradients flowing through the 'explicit' recurrent connections are ignored and only those flowing through the 'implicit' recurrence represented by the dynamics of individual neurons are considered, much more computationally tractable learning rules can be derived [26]. Learning rules of this sort include SuperSpike [25], eProp [4] and Decolle [15]. However, in order to apply these new spike-based machine learning techniques to larger models and data-sets as well as prototyping algorithms for neuromorphic hardware [8, 11, 18], new tools are required which can efficiently simulate SNNs on existing hardware. The development of efficient SNN simulators has been a key area of computational neuroscience research for several decades [1, 6, 12, 13, 23] but, these simulators are not well-suited to the types of model and the workflows required for spike-based machine learning research. As such, many ML researchers have chosen to build libraries [9, 10, 14, 19] on top of more familiar tools such as PyTorch. However, while libraries like PyTorch are highly-optimised for rate-based models, they does not take advantage of the spatio-temporal sparsity of SNNs which have the potential to enable massive computational savings over rate-based networks [24].

While our GeNN simulator [16, 17, 23] was originally developed for Computational Neuroscience research, its longstanding focus on flexibility and its targeting of GPU accelerators has made it easily adaptable to the needs of spike-based ML. Specifically, we have added support for parallel batch simulation of models which allows multiple copies of the model to be run simultaneously to maximise GPU occupancy. We have also added support for user-defined "custom update" operations which can be used to implement a wide range of functionality including gradient-based optimizers, efficient matrix transpose operations and the reduction of variables across batches. In this paper we demonstrate how these new extensions can be used to efficiently implement and train SNN classifiers using eProp [4] on the Spiking Heidelberg Digits [7] and the Spiking Sequential MNIST [20] datasets.

2 RESULTS

We trained LSNNs of various sizes with feedforward and recurrent connectivity on the Spiking Heidelberg Digits (SHD) [7] and spiking sequential MNIST [20] datasets using eProp with the default parameters Bellec et al. [4]. When comparing the performance of models trained with eProp on the SHD dataset with results obtained using BPTT [27] we found no consicerable differences for the 256 neuron models (figure 1). However, because the reduced memory requirement of eProp allows training larger models, we were able

NICE '22, March 28–April 1, 2022, San Antonio, TX

ACM ISBN 9978-1-4503-9559-5…$15.00
https://doi.org/10.1145/1122445.1122456

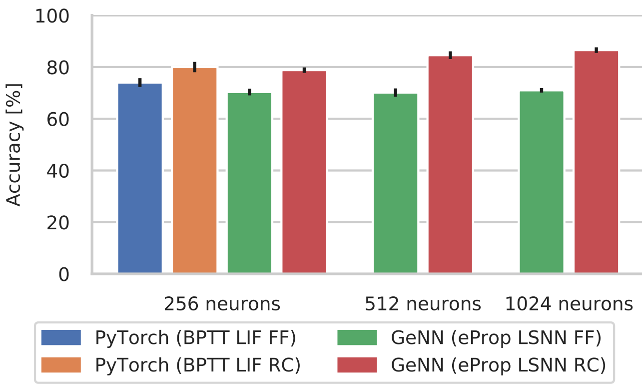

Figure 1: Performance comparison of LSNNs trained with eProp using GeNN and SNNs trained with BPTT using PyTorch [27] on Spiking Heidelberg Digits dataset. Bars signify the mean and error bars the standard deviation over 5 (GeNN) and 10 (PyTorch) simulations.

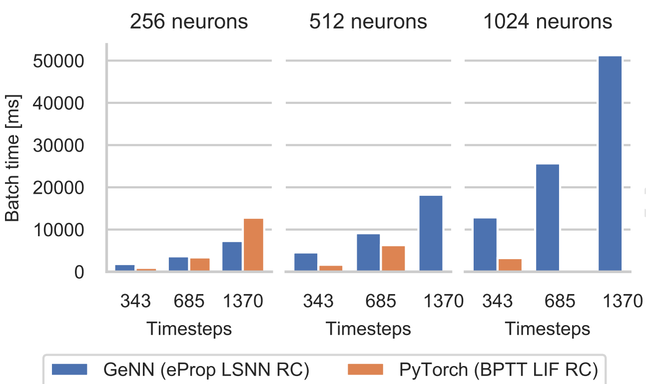

Figure 2: Training times comparison of LSNNs trained with eProp using GeNN and SNNs trained with BPTT using PyTorch on Spiking Heidelberg Digits dataset. A 12 GB Titan V GPU is used for all experiments. Input spike trains are binned to achieve different numbers of timesteps. Missing bars indicate insufficient memory for experiment.

to discover that the performance can be significantly improved by increasing the number of neurons to 512.

While being able to train networks with high classification accuracy is important, training also needs to be fast. Figure 2 shows how the time taken to train recurrent LSNNs using eProp with GeNN compares against training recurrent LIF networks using BPTT with PyTorch. For short input sequences, training using GeNN is slower (probably due to PyTorch's use of tensor cores) but, as sequence length increases, GeNN runtime increases slower and the memory requirements of BPTT mean that PyTorch is unable to train models larger than 256 neurons on 1370 timestep input sequences.

Figure 3 compares the inference time and energy delay products (EDPs) of LSNNs simulated with GeNN against LSTM models running on the same hardware [20] as well as LSNNs running on the Loihi neuromorphic chip [8]. On the same Titan V GPU, LSNNs

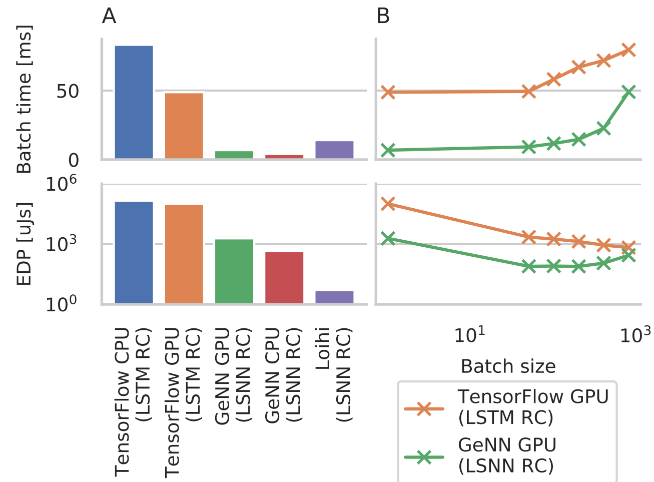

Figure 3: Comparison of inference time and energy delay products of LSNNs and LSTMs. (A) Batch size 1 including Loihi data from Plank et al. [20]. (B) Across batch sizes.

are faster than LSTMs and have a lower EDP across all batch sizes. Compared to using LSTMs, LSNN inference has a much lower arithmetic intensity meaning that, at batch size 1, not only is the CPU code generated by GeNN faster than TensorFlow running on CPU but it is also faster than GeNN running on GPU. Finally, although LSNN inference on Loihi has a much lower Energy-Delay Product, inference on both GPU and CPU using GeNN has lower latency.

3 CONCLUSIONS

By adding additional functionality aimed at accelerating spike-based machine learning workflows to our GeNN simulator, we have demonstrated that training using 'forward-mode' learning rules like eProp can not only result in competitive accuracy in classification tasks but also allow larger models to be trained on longer input sequences than is possible when using BPTT. We demonstrate that, by exploiting temporal sparsity, standard CPU and GPU hardware can perform inference faster and with less energy using LSNNs than it can using standard LSTM models. However, the eProp learning rule requires time-driven updates which dominate the time taken to *train* these models. Therefore, we are working to implement the fully event-driven EventProp [22] learning rule in GeNN which will allow training times to also benefit from temporal sparsity. Finally, the models presented in this paper are all densely connected so are not taking advantage of connection sparsity. We are working in parallel to address this by combining these learning rules with the Deep-R [2] rewiring rule, enabling SNN classifiers to take advantage of GeNN's support for efficient sparse connectivity [17].

ACKNOWLEDGMENTS

This work was funded by the EPSRC (grant numbers EP/P006094/1 and EP/S030964/1) and the EU's Horizon 2020 program (grant agreement 945539). Compute time was provided through Gauss Centre for Supercomputing application number 21018 and EPSRC (grant number EP/T022205/1).

REFERENCES

[1] Nora Abi Akar, Ben Cumming, Vasileios Karakasis, Anne Kusters, Wouter Klijn, Alexander Peyser, and Stuart Yates. 2019. Arbor âĂŤ A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures. In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 274–282. https://doi.org/10.1109/EMPDP.2019.8671560 arXiv:1901.07454

[2] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. 2018. Deep rewiring: Training very sparse deep networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (2018), 1–24. arXiv:1711.05136

[3] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, Vol. 2018-Decem. 787–797. arXiv:1803.09574

[4] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. 2020. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications* 11, 1 (dec 2020), 3625. https://doi.org/10.1038/s41467-020-17236-y

[5] Sander M. Bohte. 2011. Error-backpropagation in networks of fractionally predictive spiking neurons. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6791 LNCS, PART 1 (2011), 60–68. https://doi.org/10.1007/978-3-642-21735-7_8

[6] Nicholas T Carnevale and Michael L Hines. 2006. *The NEURON book*. Cambridge University Press.

[7] Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. 2020. The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020). https://doi.org/10.1109/TNNLS.2020.3044364 arXiv:1910.07407

[8] Mike Davies, Narayan Srinivasa, Tsung-han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steve Mccoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi : a Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 30, 1 (2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[9] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. 2021. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:1906.09395* (2021).

[10] Wei Fang, Yanqi Chen, Jianhao Ding, Ding Chen, Zhaofei Yu, Huihui Zhou, Yonghong Tian, and other contributors. 2020. SpikingJelly. https://github.com/fangwei123456/spikingjelly.

[11] Stephen B Furber, Francesco Galluppi, S Temple, and Luis A Plana. 2014. The SpiNNaker Project. *Proc. IEEE* 102, 5 (may 2014), 652–665. https://doi.org/10.1109/JPROC.2014.2304638

[12] Marc-Oliver Gewaltig and Markus Diesmann. 2007. NEST (NEural Simulation Tool). *Scholarpedia* 2, 4 (2007), 1430.

[13] Bruno Golosio, Gianmarco Tiddia, Chiara De Luca, Elena Pastorelli, Francesco Simula, and Pier Stanislao Paolucci. 2021. Fast Simulations of Highly-Connected Spiking Cortical Models Using GPUs. *Frontiers in Computational Neuroscience* 15, February (feb 2021), 1–17. https://doi.org/10.3389/fncom.2021.627620

[14] Hananel Hazan, Daniel J. Saunders, Hassaan Khan, Devdhar Patel, Darpan T. Sanghavi, Hava T. Siegelmann, and Robert Kozma. 2018. BindsNET: A Machine Learning-Oriented Spiking Neural Networks Library in Python. *Frontiers in Neuroinformatics* 12, December (dec 2018), 1–18. https://doi.org/10.3389/fninf.2018.00089 arXiv:1806.01423

[15] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. 2020. Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE). *Frontiers in Neuroscience* 14, May (may 2020), 1–11. https://doi.org/10.3389/fnins.2020.00424 arXiv:1811.10766

[16] James C Knight, Anton Komissarov, and Thomas Nowotny. 2021. PyGeNN: A Python Library for GPU-Enhanced Neural Networks. *Frontiers in Neuroinformatics* 15, April (apr 2021). https://doi.org/10.3389/fninf.2021.659005

[17] James C. Knight and Thomas Nowotny. 2018. GPUs Outperform Current HPC and Neuromorphic Solutions in Terms of Speed and Energy When Simulating a Highly-Connected Cortical Model. *Frontiers in Neuroscience* 12, December (2018), 1–19. https://doi.org/10.3389/fnins.2018.00941

[18] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D Flickner, William P Risk, Rajit Manohar, and Dharmendra S Modha. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673. https://doi.org/10.1126/science.1254642

[19] Christian Pehle and Jens Egholm Pedersen. 2021. *Norse - A deep learning library for spiking neural networks*. https://doi.org/10.5281/zenodo.4422025 Documentation: https://norse.ai/docs/.

[20] Philipp Plank, Arjun Rao, Andreas Wild, and Wolfgang Maass. 2021. A Long Short-Term Memory for AI Applications in Spike-based Neuromorphic Hardware. (2021). arXiv:2107.03992 http://arxiv.org/abs/2107.03992

[21] Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation* 1, 2 (1989), 270–280. https://doi.org/10.1162/neco.1989.1.2.270

[22] Timo C Wunderlich and Christian Pehle. 2021. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports* 11, 1 (dec 2021), 12829. https://doi.org/10.1038/s41598-021-91786-z

[23] Esin Yavuz, James Turner, and Thomas Nowotny. 2016. GeNN: a code generation framework for accelerated brain simulations. *Scientific Reports* 6, 1 (may 2016), 18854. https://doi.org/10.1038/srep18854

[24] Bojian Yin, Federico Corradi, and Sander M. Bohte. 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. (2021), 1–15. arXiv:2103.12593 http://arxiv.org/abs/2103.12593

[25] Friedemann Zenke and Surya Ganguli. 2018. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation* 30, 6 (jun 2018), 1514–1541. https://doi.org/10.1162/neco_a_01086 arXiv:NIHMS150003

[26] Friedemann Zenke and Emre O. Neftci. 2021. Brain-Inspired Learning on Neuromorphic Substrates. *Proc. IEEE* (2021), 1–16. https://doi.org/10.1109/JPROC.2020.3045625 arXiv:2010.11931

[27] Friedemann Zenke and Tim P. Vogels. 2021. The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. *Neural computation* 33, 4 (2021), 899–925. https://doi.org/10.1162/neco_a_01367