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 SNNs on GPUs for computational neuroscience, we are extending our GeNN SNN simulator to be a competitive tool for spike-based machine learning research on general purpose hardware. To explore GeNN's potential, we train SNN classifiers on the Spiking Heidelberg Digits and the Spiking Sequential MNIST datasets using the eProp learning rule. We find that the performance of these classifiers is comparable 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

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
James C Knight and Thomas Nowotny. 2021. Efficient GPU training of LSNNs using eProp. In *NICE '21: 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, a plethora of new techniques for directly training spiking neural networks (SNNs) using gradient-based approaches have been developed. One such approach is to replace the non-differentiable 'transfer function' of a spiking neuron with a surrogate, allowing an SNN to be trained using the same algorithms used to train rate-based Recurrent Neural Network (RNNs) such as Back Propagation Through Time (BPTT) [2, 4, 19]. 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 [15] 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 flow of gradients through the 'explicit' recurrent connections is ignored and only those flowing through the 'implicit' recurrence within each spiking neuron are considered, much more computationally tractable learning rules can be derived [18]. Learning rules of this sort include SuperSpike [17], eProp [2] and Decolle **(TODO: cite)**.

In order to apply new spike-based machine learning techniques to larger models and data-sets as well as prototyping algorithms for neuromorphic hardware **(TODO: cite)**, 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, 5, 10, 11, 16] but, these simulators **(TODO: more)**, making them ill-suited for spike-based machine learning research. As such, many ML researchers have chosen to build libraries [8, 9, 12, 13] on top of more familiar tools such as PyTorch. However, while PyTorch is highly-optimised for rate-based models, it does not take advantage of the spatio-temporal sparsity of SNNs which have the potential to enable massive computational savings over rate-based networks **(TODO: cite sander)**. GENN Specialised neuromorphic hardware **(TODO: cite usual suspects)** can be used to implement SNNs very efficiently but, while some consensus is beginning to emerge on the theoretical framework within which biological learning may operate, casting algorithms into hardware always results in a game of catch-up with even the most flexible current neuromorphic systems being capable of implementing only very limited versions of the latest online learning rules **(TODO: cite garrick)**.

2 RESULTS

We trained LSNNs of various sizes with both feedforward and recurrent connectivity on the Spiking Heidelberg Digits (SHD) [6] and spiking sequential MNIST [14] datasets using eProp with the default parameters employed by Bellec et al. [3].

2.1 Accuracy

In figure 2 we compare the performance of models trained on the SHD dataset with results obtained by Zenke and Vogels [19]. There is no significant **(TODO: suitable significance test?)** different between the performance of the 256 neuron models and, because the reduced memory requirement of our ePropimplementation allows larger models to be trained, we show that the performance can be significantly improved by increasing the number of neurons to 512.

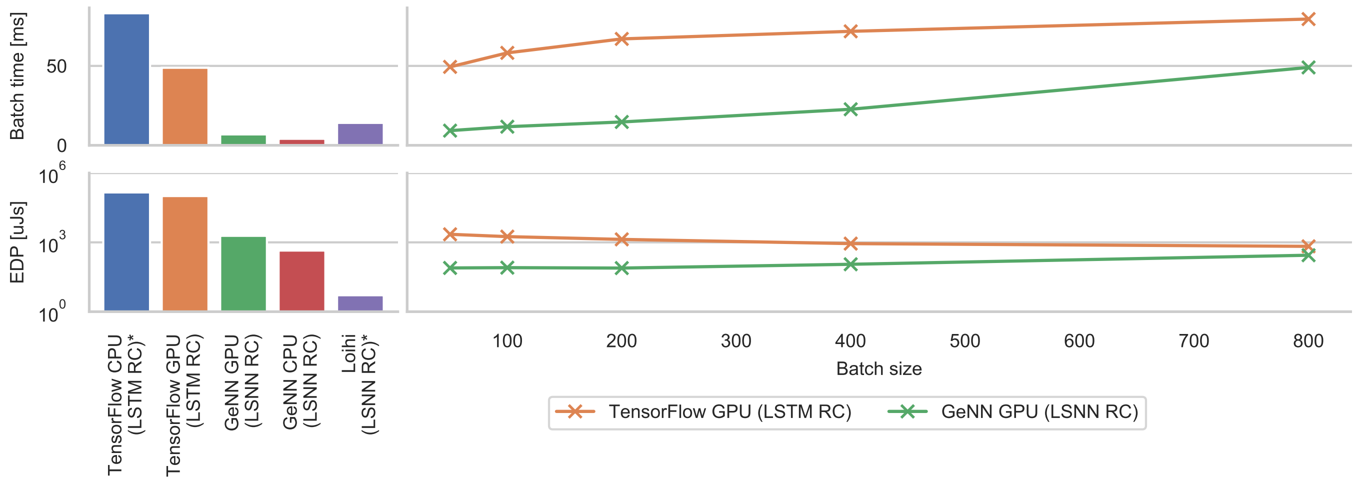

Figure 1: 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

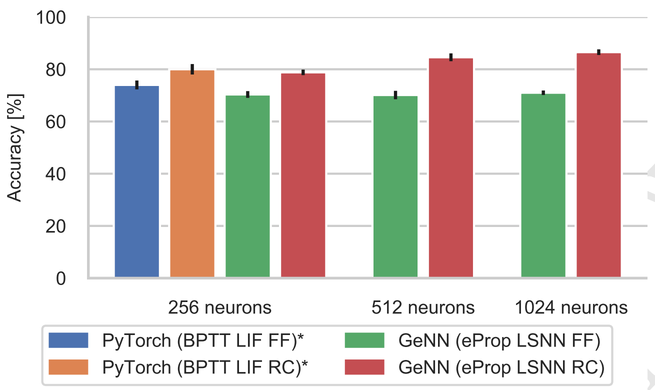

Figure 2: Comparison of performance on Heidelberg Spiking Digits datasets of LSNNs trained with eProp using GeNN and SNNs trained with BPTT using PyTorch [19]. Bars signify the mean and error bars the standard deviation over 5 (GeNN) and 10 (PyTorch) simulations.

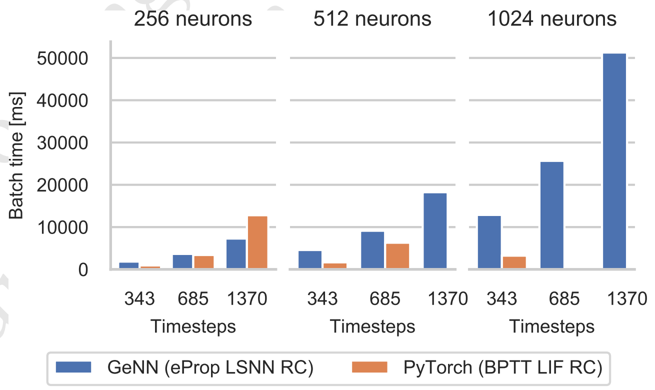

Figure 3: 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

2.2 Training time

While being able to train networks with high classification accuracy is important, training also needs to fast. In figure 3 we compare the time taken to train recurrent LSNNs using eProp with GeNN against recurrent LIF networks training using BPTT with PyTorch.

2.3 Inference time

In figure 1, we compare the inference time and energy delay products **(TODO: CITE)** of LSNNs simulated using GeNN against LSTM models running on the same hardware [14] as well as LSNNs running on the Loihi neuromorphic system [7]. On the same Titan V GPU, LSNNs are faster than LSTMs and have a lower EDP across all batch sizes although, at the largest batch sizes, the gap reduces. When using batch size 1, the gap between Compared to inference using LSTMs, LSNN inference has a much lower arithmetic intensity meaning that it is much better suited to CPU implementation. Finally, although LSNN inference on Loihi has a much lower EDP, inference on both GPU and CPU using GeNN has lower latency.

3 CONCLUSIONS

By adding additional functionality aimed at accelerating common machine learning workflows to our GeNN simulator, we have demonstrated that training using forward-mode "approximate RTRL" learning rules like eProp can not only result in competitive accuracy in classification tasks but also that larger models can be trained on longer stimuli than is possible when using BPTT. Finally 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.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

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, 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

[3] 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

[4] 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

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

[6] 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

[7] 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, Yihsin 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

[8] 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).

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

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

[11] 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

[12] 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

[13] 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/.

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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