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, ?? pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, several new techniques for directly training spiking neural networks (SNNs) have been developed. One approach is to replace the non-differentiable 'transfer function' of a spiking neuron with a differentiable surrogate function [? ? ?], 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 [?] 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 [?]. Learning rules of this sort include SuperSpike [?], eProp [?] and Decolle [?]. **(TODO: some background on spike-time based learning)** 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 [? ? ?], 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 [? ? ? ? ?] 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 [? ? ? ?] 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 [?].

While our GeNN simulator [? ? ?] was originally developed for Computational Neuroscience research, its longstanding focus on flexibility and its targeting of GPU accelerators make it well-suited to the needs of spike-based ML. **(TODO: talk about additions to GeNN)** Here we demonstrate this by training SNN classifiers on the Spiking Heidelberg Digits [?] and the Spiking Sequential MNIST [?] datasets.

2 RESULTS

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

When comparing the performance of models trained with eprop on the SHD dataset with results obtained using BPTT ?] we found no significant differences **(TODO: suitable significance test?)** for the 256 neuron models (figure ??). However, because the reduced memory requirement of eProp allows training larger models, we were able 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 fast. Figure ?? shows how the time taken to train recurrent LSNNs using eProp with GeNN compares against recurrent LIF networks training using BPTT with PyTorch. When using short input sequences, training using GeNN is slower (probably due to PyTorch's use of our GPU's tensor cores)

NICE '21, March 28–April 1, 2022, San Antonio, TX

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

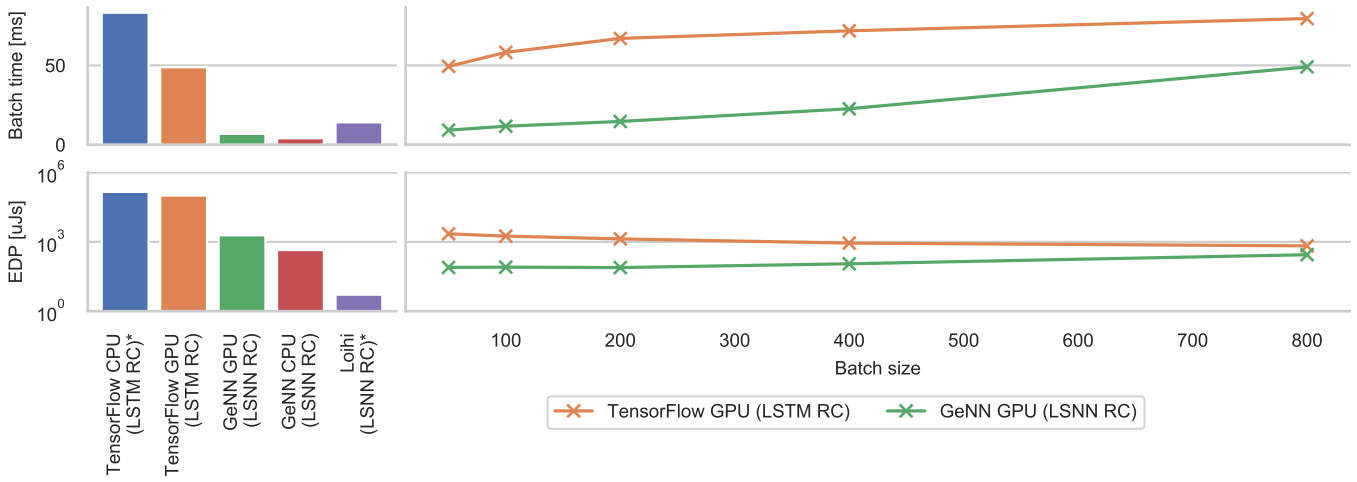

Figure 1: 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

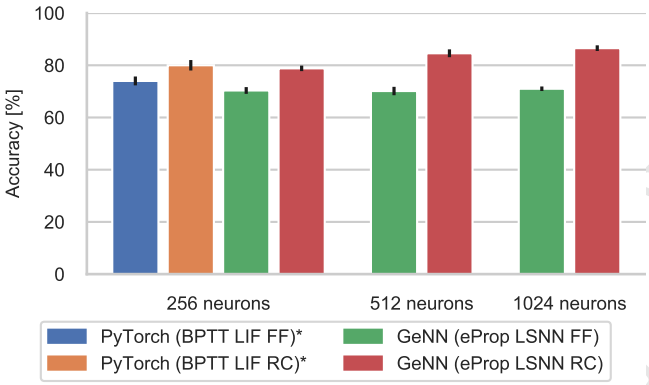

Figure 2: Performance comparison of LSNNs trained with eProp using GeNN and SNNs trained with BPTT using PyTorch [?] on Spiking Heidelberg Digits dataset. Bars signify the mean and error bars the standard deviation over 5 (GeNN) and 10 (PyTorch) simulations.

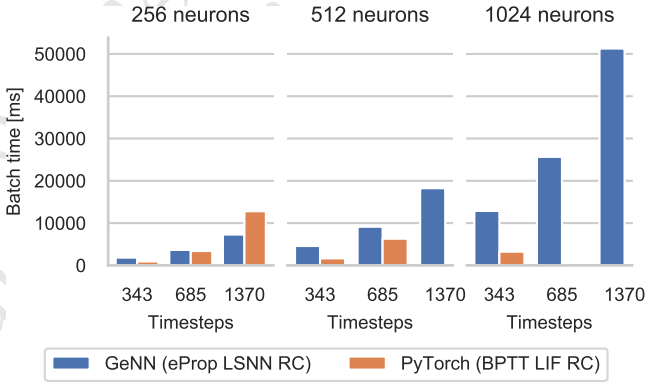

Figure 3: Training times comparison of LSNNs trained with eProp using GeNN and SNNs trained with BPTT using PyTorch on Spiking Heidelberg Digits dataset. A 12 GB Titan V GPU is used for all experiments. Input spike trains are binned to achieve different numbers of timesteps. Missing bars indicate insufficient memory for experiment.

but, as sequence length increases, GeNN runtim 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.

In figure ??, we compare the inference time and energy delay products **(TODO: CITE)** of LSNNs simulated using GeNN against LSTM models running on the same hardware [?] as well as LSNNs running on the Loihi neuromorphic system [?]. On the same Titan V GPU, LSNNs 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 Prodyct, inference on both GPU and CPU using GeNN has lower latency.

3 FUTURE WORK

(TODO: EventProp, convolutional networks, deeper networks)

4 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. 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.