Low-Power Deep Learning Inference using the SpiNNaker Neuromorphic Platform

Craig M. Vineyard, Ryan Dellana, James B. Aimone, William M. Severa
Sandia National Laboratories
Albuquerque, NM
cmviney@sandia.gov,rdellan@sandia.gov,jbaimon@sandia.gov,wmsever@sandia.gov

ABSTRACT

In this presentation we will discuss recent results on using the SpiNNaker neuromorphic platform (48-chip model) for deep learning neural network inference. We use the Sandia Labs developed Whetstone spiking deep learning library to train deep multi-layer perceptrons and convolutional neural networks suitable for the spiking substrate on the neural hardware architecture. By using the massively parallel nature of SpiNNaker, we are able to achieve, under certain network topologies, substantial network tiling and consequentially impressive inference throughput. Such high-throughput systems may have eventual application in remote sensing applications where large images need to be chipped, scanned, and processed quickly. Additionally, we explore complex topologies that push the limits of the SpiNNaker routing hardware and investigate how that impacts mapping software-implemented networks to on-hardware instantiations.

1 INTRODUCTION & BACKGROUND

Historical advances in computational capabilities have been enabled by trends such as Moore's Law and Dennard Scaling. With these scaling laws coming to an end, rather than exponential advances in manufacturing capability furthering general purpose architectures instead specialized architectures are being considered. In particular, the growing popularity of deep neural networks and machine learning is fueling research into accelerators better suited for these workloads [2]. With the proliferation of machine learning accelerators and neuromorphic architectures, there has been a lag in the development of the software stack to interface to these emerging devices. The Neural Network Exchange Format provides a specification in which emerging architectures can interface to by providing parsing and compilation tools specific to their architecture [5]. Similarly, Glow developed by Facebook strives to provide an intermediate representation of high level machine learning representations for which hardware targets can support [8]. These and other emerging efforts are not presently targeting spiking neuromoprhic hardware, and additionally the deep neural network algorithms themselves need to either learn spiking representations or be converted to map to neuromorphic hardware.

Developed by Sandia Labs, Whetstone is a method for training deep artificial neural networks for binary communication [10, 11]. The whetstone method, which was presented at NICE 2018, offers an iterative approach that integrates well with a variety of neural network topologies and integrates with standard deep learning libraries (namely Keras on Tensorflow). The Whetstone method is illustrated in Fig. 1A. This allows us to quickly prototype and train deep spiking neural networks that are neuromorphic-ready at a variety of scaled and complexities. Using Whetstone to target neuromorphic architectures, next will present results on the University of Manchester SpiNNaker architecture.

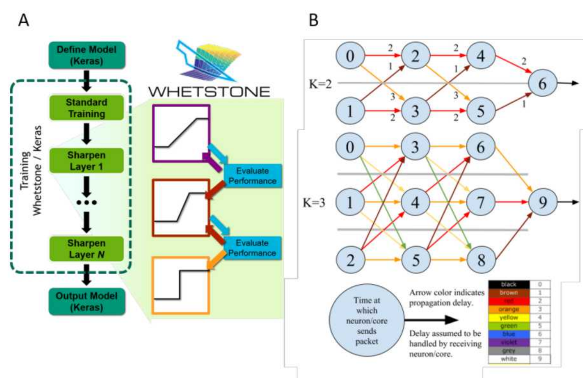

Figure 1: A) Illustration of the Whetstone Method B) Depiction of the temporal distribution of spikes to avoid communication overloads on SpiNNaker

2 RESULTS

The SpiNNaker architecture is a scalable, energy-efficient, massively parallel computing architecture inspired by the function and structure of the brain [3, 4]. The fundamental unit of the SpiNNaker architecture is a multicore System-on-Chip comprised of lightweight ARM968 processor cores each with local instruction and data memory, and a packet router. Multi-cast routing via address event representation both allows cores within each chip to communicate as well as chips within the overarching architecture. A communication mesh allows spikes from individual neurons to route to any other neuron in the system. For our work here, we have used a 48-node (864 core) SpiNNaker board.

For mapping applications onto the SpiNNaker infrastructure, low level C or Python code may be used as well as high-level neural description languages such as PyNN [1] and Nengo [12]. SpyNNaker provides a software package for simulating PyNN defined networks on the SpiNNaker platform [7].

2.1 Mapping

Given the available software interfaces to the SpiNNaker platform, we chose to implement Whetstone networks using SpyNNaker. This interface provides a higher-level abstraction compared to the C interface, and we expect, going forward, that SpyNNaker will provide greater interoperability with a range of peripheral libraries

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

(data loaders, spiking sensors, etc.). Given the flexibility of connections, most components and parameters of the Whetstone network map directly to the SpyNNaker network. However, we have noticed several important details:

(1) **Bias** Most deep neural network use a bias that helps improve separability away from the origin. In a Whetstone network, the bias is trained as with any other weight parameter. However, in practice, the bias is 'rolled' into the threshold of the neuron. For SpyNNaker networks, this implies a different threshold for each neuron, which by default, would require each neuron to be on separate cores. We alleviate this problem by either (1) training a network without bias or (2) using a separate neuron to relay a bias signal (via weights). This signal allows the neurons to each have an effective threshold without changing the threshold parameter.

(2) **Neuron Types** Whetstone networks require very simple neurons (essentially perceptrons). In contrast, SpyNNaker (and SpiNNaker) can support much more complicated neuron models. We use the IF_curr_delta neuron model in standard implmentation, but we have also created a custom neuron type IF0_curr_delta that allows denser neuron packing.

(3) **Weight Precision** Since Whetstone uses standard deep learning methods, the weights are floating point parameters (either 16-bit or 32-bit). However, most neuromorphic platforms use some fixed-point representation; SpiNNaker uses $Q15.16$ integer values. As such, we decided to investigate the impact of reduced fixed-point precision weights. In general, we have seen that Whetstone networks remain effective, with essentially no loss in performance, down to $Q4.8$.

2.2 Communication & Throughput

We have explored implementing small dense multi-layer neural networks as well as a convolutional neural network distributed across the SpiNNaker 48 platform. Given the scale of the platform and the small footprint of basic MLPs, we anticipate an optimal 'accuracy/throughput' configuration to involve network tiling—where multiple copies of the network are instantiated across different chips/cores and multiple inputs are fed simultaneously onto the board. This configuration is compatible with many image-scanning applications such as those involving satellite imagery and remote area mapping.

During testing, we did notice that having a large number of neurons (≈ 400) spike simultaneously within a core overwhelms the SpiNNaker router, causing errors. While a small number of dropped or delayed spikes may be tolerable for the network, we still wanted to develop methods to mitigate this hardware-specific issue. For any layer configuration (i.e., convolution or densely connected), we can use the flexible delays supported in-hardware to temporally demux the signal, see Fig. 1B. More specifically, we split a layer into different temporal groups each of which have outbound (axon-side) synapses of a different delay. Then, post-synaptic temporal groups integrate signals distributed in time, which lowers the number of spikes generated at any given timestep. However, care must be taken to ensure that the correct spikes are delivered to downstream neurons in-sync. For convolutional layers, we recognize that the connectivity is spatially oriented. As such, we can distribute neurons across cores so that nearby neurons (regardless of layer) are located in the same core. Preliminary analysis suggests this improves communication bottlenecks, but requires careful layout. We are still investigating this method of distributing networks and hope to have full results soon.

Table 1 provides the results and paramaterization of applying three different neural network topologies to the Binary Mnist dataset. The network toplogies are as follows with 10-hot output encoding:

(1) Small MLP: 785 → 100 → 100 → 100
(2) Medium MLP: 785 → 800 → 800 → 800 → 100
(3) Convolution Network: 785 → conv1 ($5x5$ $x32$ kernels) → maxpool (2,2) → conv2 ($5x5$ $x64$ kernels) → maxpool (2,2) → 500 dense → 100

Table 1: Performance of parallel networks on SpinNNaker 48

Network	Small MLP	Medium MLP	Convolution Network
Total Neurons	57000	72500	47640
Total Cores	760	754	371
Total Chips Utilized	48	48	28
Network Tiles	190	29	1
Timescale Factor	5.0	6.0	14.0
Sample Delay (ms)	2	2	28
Throughput (frames/sec)	15317	2340	3.25
Accuracy	94%	97.7%	98.1%

3 CONCLUSIONS & FUTURE WORK

These results show the impact of Whetstone trained binary communication neural networks exploiting the parallelism of the SpiNNaker neuromorphic platform. Previously, Serrano-Gotarrendona et al. used temporal coding to put similar sized convolutional networks on SpiNNaker [9]. And the recent work by Liu et al. explores putting deep neural networks on the SpiNNaker 2 prototype introducing DEEP R as a method to train an algorithm on board by continuously re-writing the network [6]. Our approach utilizing the Whetstone method provides an alternative method able to attain high throughput by utilizing binary communication rather than temporal coding. In doing so we are also able to achieve high classification accuracy relative to comparable spiking neuromoprhic approaches.

By distributing many copies of a network across the SpiNNaker 48 platform, this highlights the potential power to implement an ensemble of networks in parallel or to alternatively partition the processing of a larger aggregate image into smaller more manageable tiles. In this latter case, rather than having a monolithic DNN operating upon a large image, such as those generated by remote sensing satellites, smaller tiles from the overarching scene can be processed in parallel. For example, rather than 190 or 29 network tiles each processing different binary MNIST inputs as we have shown here, instead they could be sub-samples of a larger image. This shows exciting promise for a large neuromorphic platform to help enable real time processing of sensor data.

ACKNOWLEDGMENT

This work was supported by the Advanced Simulation and Computing, and the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

REFERENCES

[1] Brüderle, D., Müller, E., Davison, A. P., Muller, E., Schemmel, J., and Meier, K. Establishing a novel modeling tool: a python-based interface for a neuromorphic hardware system. *Frontiers in neuroinformatics 3* (2009), 17.

[2] Dean, J., Patterson, D., and Young, C. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro 38*, 2 (2018), 21–29.

[3] Furber, S. Large-scale neuromorphic computing systems. *Journal of neural engineering 13*, 5 (2016), 051001.

[4] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. Overview of the spinnaker system architecture. *IEEE Transactions on Computers 62*, 12 (2013), 2454–2467.

[5] Group, T. K. N. W. Neural network exchange format version 1.0 revision 3, 2018.

[6] Liu, C., Bellec, G., Vogginger, B., Kappel, D., Partzsch, J., Höppner, S., Maass, W., Furber, S. B., Legenstein, R., Mayr, C. G., et al. Memory-efficient deep learning on a spinnaker 2 prototype. *Frontiers in Neuroscience 12* (2018), 840.

[7] Rhodes, O., Bogdan, P. A., Brenninkmeijer, C., Davidson, S., Fellows, D., Gait, A., Lester, D. R., Mikaitis, M., Plana, L. A., Rowley, A. G., et al. spynnaker: a software package for running pynn simulations on spinnaker. *Frontiers in Neuroscience 12* (2018), 816.

[8] Rotem, N., Fix, J., Abdulrasool, S., Deng, S., Dzhabarov, R., Hegeman, J., Levenstein, R., Maher, B., Nadathur, S., Olesen, J., et al. Glow: Graph lowering compiler techniques for neural networks. *arXiv preprint arXiv:1805.00907* (2018).

[9] Serrano-Gotarredona, T., Linares-Barranco, B., Galluppi, F., Plana, L., and Furber, S. Convnets experiments on spinnaker. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on* (2015), IEEE, pp. 2405–2408.

[10] Severa, W., Vineyard, C. M., Dellana, R., Verzi, S. J., and Aimone, J. B. Whetstone: A method for training deep artificial neural networks for binary communication. *arXiv preprint arXiv:1810.11521* (2018).

[11] Severa, W. M., Vineyard, C. M., Dellana, R., and Aimone, J. B. Whetstone: An accessible, platform-independent method for training spiking deep neural networks for neuromorphic processors. In *SysML Conference* (2018).

[12] Stewart, T. C., and Eliasmith, C. Large-scale synthesis of functional spiking neural circuits. *Proceedings of the IEEE 102*, 5 (2014), 881–898.