Whetstone: An accessible, platform-independent method for training spiking deep neural networks for neuromorphic processors

William M. Severa
wmsever@sandia.gov
Sandia National Laboratories
Albuquerque, NM

Craig M. Vineyard
cmviney@sandia.gov
Sandia National Laboratories
Albuquerque, NM

Ryan Dellana
rdellan@sandia.gov
Sandia National Laboratories
Albuquerque, NM

James B. Aimone
jbaimon@sandia.gov
Sandia National Laboratories
Albuquerque, NM

ACM Reference Format:
William M. Severa, Craig M. Vineyard, Ryan Dellana, and James B. Aimone. 2018. Whetstone: An accessible, platform-independent method for training spiking deep neural networks for neuromorphic processors. In *Proceedings of SysML.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

We present a method wherein a sequence of convergent activation functions enable high-speed training of deep spiking neural networks. This method is accessible, easily implementable and generalizable to a variety of network structures. By democratizing the use of neuromorphic processors for deep learning, the Whetstone method enables non-neuromorphic-computing experts to use these cutting-edge platforms and achieve corresponding improvements in performance-per-Watt.

Current image processing methods are largely dominated by deep neural networks and, in particular, convolutional neural networks. While machine learning methods have even surpassed human performance in some tasks [5], they run best on power-hungry GPUs [1]. In contrast, there are a growing number of biologically inspired neuromorphic architectures offering dramatic improvements in performance-per-Watt compared to GPUs [3, 6, 8, 9]. However, it is often difficult to achieve state-of-the-art algorithmic performance using spiking neural networks (SNNs), and few of these platforms provide learning on-chip.

The Whetstone method bridges these two spaces by iteratively converging towards a spiking version of a deep neural network. During the training process, the activation function at each layer is progressively sharpened towards a threshold activation. This method is successful with both sigmoid activations and bounded rectified linear units. Since we only need to modify the activation function, Whetstone is widely applicable to a range of network designs and connectivitiy (e.g. densely connected layers, convolutional layers). We have implemented Whetstone as a package extending the Keras deep learning library [2]. By implementing Whetstone as a variety of custom components within Keras, we achieve a drop-in workflow.

SysML, February 15–16, 2018, Stanford, CA, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The overall task is the same as that of any neural network, and we abstract this task as evolving weights W_t such that $f(W_t\mathbf{x}) \to_t \hat{\mathbf{y}}$, where $\mathbf{x}$ is the input set, f is a non-linear activation function and $\hat{\mathbf{y}}$ is the best local prediction (subject to some loss function) of the ground truth $\mathbf{y}$. In traditional neural networks, the function f generally has a simple gradient to assist fast convergence using backprpopogation and a stochastic gradient descent algorithm. However, these activation functions are incompatible with the spiking neuron models used by most neuromorphic platforms, and spiking neurons offer distinct power advantages, see [4, 8, 9]. The Whetstone method is centered on using a sequence of convergent activation functions f_k, each one continuous, that approach (in measure) the target threshold activation function $f_k \to_k T(x) := x \geq 0$. In short, we are interested in $\lim_{k,t} f_k(W_t\mathbf{x})$. Unfortunately, since in general the convergence in neither t nor k is uniform, the interchange of limits is poorly defined. Though, through experimentation and heuristics, we have seen strong performance in a variety of tasks and network structures.

For experimentation and initial release, we implemented Whetstone as a python package extending Keras. Whetstone spiking activation functions are drop-in replacements for the built-in activation functions, and a sharpening callback automatically iterates the activation functions towards a threshold activation (Fig. 2). After training, the weights are directly exportable. Interestingly, in our experiments we observed trade-offs not usually seen in conventional deep neural networks (DNNs). Unsurprisingly this suggests that best-practices for network design may differ between DNNs and SNNs. Some compelling observations include:

- Larger convolutional filters (e.g. 5×5 or 7×7) improve performance and stability of training.
- Resilient, fault-tolerant output codings are critical.
- Sigmoid activation, while still suffering vanishing gradient, provides increased stability. Modified rectified linear units provide the best network performance.
- The process is sensitive to the choice of optimizer. As examples, RMSProp [10] and Adadelta [11] offer fast and reliable convergence. Other standard optimizers (e.g. Adam [7]) are unstable.

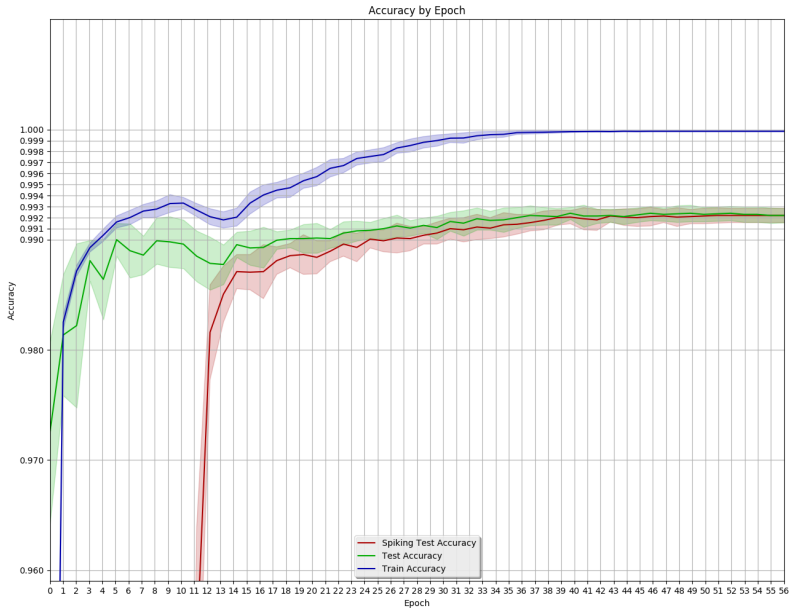

Figure 1: Training, testing and spiking testing accuracy for a standard deep convolutional network on the MNIST dataset. The network has six convolutional layers and 11 total layers.

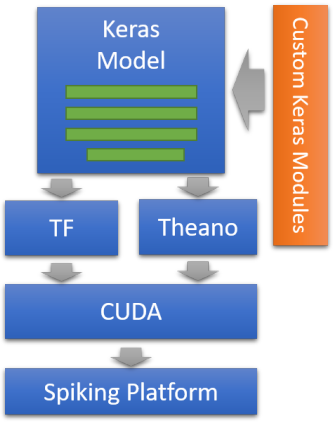

Figure 2: A schematic of Whetstone's python implementation. By interjecting at the same level of abstraction as Keras, we leverage existing work in Tensorflow, Theano, CUDA, and the target platform.

This approach provides various advantages over other methods for designing or training SNNs. Foremost, the method is quick, accessible, and platform-independent. Each layer involves one timestep, so no additional time cost is incurred as in rate and temporal coding. The requirements on the target neuron model are simple (integrate and fire). Hence, resulting networks have wide applicability. And while, in general, a spiking version of a neural network may show decreased performance, most algorithms remain competitive with approximately 0.5% loss in accuracy. For example, with a standard network design we reach an overall maximum MNIST accuracy of 99.37% max (99.22% median) directly (i.e. no ensemble methods, no data augmentation), see Figure 1.

ACKNOWLEDGEMENTS

Sandia National Laboratories is a multimission 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-NA0003525.

REFERENCES

[1] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf* (2010), pp. 1–7.

[2] Chollet, F., et al. Keras. https://github.com/fchollet/keras, 2015.

[3] Esser, S. K., Appuswamy, R., Merolla, P., Arthur, J. V., and Modha, D. S. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1117–1125.

[4] Hasler, J., and Marr, B. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in neuroscience 7* (2013).

[5] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)* (December 2015).

[6] Hill, A. J., Donaldson, J. W., Rothganger, F. H., Vineyard, C. M., Follett, D. R., Follett, P. L., Smith, M. R., Verzi, S. J., Severa, W., Wang, F., Aimone, J. B., Naegle, J. H., and James, C. D. A spike-timing neuromorphic processor. In *Proceedings of the IEEE Conference on Rebooting Computing* (2017).

[7] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] Schemmel, J., Bruderle, D., Grubl, A., Hock, M., Meier, K., and Millner, S. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on* (2010), IEEE, pp. 1947–1950.

[9] Stromatias, E., Galluppi, F., Patterson, C., and Furber, S. Power analysis of large-scale, real-time neural networks on spinnaker. In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–8.

[10] Tieleman, T., and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4*, 2 (2012), 26–31.

[11] Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).