Whetstone
An Accessible, Platform-Independent Method for Training Spiking Deep Neural Networks for Neuromorphic Processors

William M. Severa*, Craig M. Vineyard, Ryan Dellana and James B. Aimone
What is missing for neuromorphic to go mainstream?

Neuromorphic hardware is
- Available
- Competitive
- Constantly improving
What is missing for neuromorphic to go mainstream?

Accessibility

Algorithms

Applications
Introduction

Deep Learning Stack

- Frameworks
  - TensorFlow
  - PyTorch
  - Keras

Platforms

- TVM

Hardware

Neuromorphic Stack

- Interfaces
  - N2A
  - IBM

Neuromorphic
Whetstone **Overview**

Whetstone provides a drop-in mechanism for tailoring a DNN to a spiking hardware platform (or other binary threshold activation platforms)

- Hardware platform agnostic
- Compatible with a wide variety of DNN topologies
- No added time or complexity cost at inference
- Simple neuron requirements: Integrate and fire
The real challenge for deep learning on spiking is the threshold activation function.

Using Whetstone, activation functions converge to a threshold activation *during training*. 
Whetstone Overview

• Generally, gradient descent generates a sequence of weights $A_i$ with the goal of minimizing the error of $f(A_i x)$ in predicting the ground truth $y$.
• We generalize this by replacing the activation function $f$ with a sequence $f_k$ such that $f_k \rightarrow_{L_1} f$, where $f$ is now the threshold activation function.
• Now, the optimizer must minimize the error of $f_k(A_i x)$ in predicting $y$.
• Since the convergence in neither $i$ nor $k$ is uniform, this is a mathematically dangerous idea.
• However, with a little care and a few tricks, the method reliably converges in many cases.
Whetstone Overview

When/Where do we decide to ‘sharpen’ the activations?

1) Bottom-up Sharpening (The ‘toothpaste tube’ method)
   - Begin sharpening at the bottom layer
   - Wait until previous layer is fully sharpened
   - Increases stability of convergence

2) Adaptive Sharpening Callback
   - Hand-tuning sharpening rates is hard
   - Instead, use loss as a guide for an *adaptive sharpener*
   - Adaptive sharpener implemented as a callback automatically adjusts sharpening based on loss thresholds

```python
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Modified Model Example:
```python
model.add(Dense(256))
model.add(Spiking_BRelu())
model.add(Dense(10))
model.add(Spiking_Brelu())
Model.add(Softmax_Decode(key))
```

Modified Model Example:
```python
sharpener = AdaptiveSharpener()
model.fit(x,y, callbacks=[sharpener])
```
Preliminary Results

Filter Size
- 7x7
- 5x5
- 3x3

Network Topologies
- Dense
- Output
- Conv
- Pool
- Dense
- Output
- Conv
- Pool
- Dense
- Output
- Conv
- Pool
- Dense
- Output
- Conv
- Pool
- Dense
- Output
- Conv
- Pool
- Dense
- Output

MNIST

Spiking Accuracy vs. Non-Spiking Accuracy
Preliminary Results

Filter Size
- 7x7
- 5x5
- 3x3

Network Topologies
- Dense
- Conv
- Pool
- Dense
- Output

Spiking Accuracy vs. Non-Spiking Accuracy

Fashion MNIST
Preliminary Results

Filter Size
7x7  5x5  3x3

Network Topologies
Dense  Dense
Conv  Conv
Pool  Pool
Output  Output

Spiking Accuracy vs. Non-Spiking Accuracy for Cifar10

- Preliminary Results
Preliminary Results

Filter Size
- 7x7
- 5x5
- 3x3

Network Topologies

Cifar100

Spiking Accuracy vs. Non-Spiking Accuracy

Preliminary Results
Established Deep Learning Techniques

Advantage: We can build on existing deep learning technology.

Software:
• Keras
• Tensorflow
• Theano
• CUDA
• Endless Python Packages

Techniques:
• Dropout
• Batch Norm
• Adaptive Optimizers
• Voting Methods
Established Deep Learning Techniques

- Batch Normalization helps training stability and network performance
- Improvements across network sizes
- Sharpnening loss, particularly on first sharpening layer, is significantly less
- At inference time, bias (threshold) and weights are modulated according to stats collected during training
Established Deep Learning Techniques

- Sharpening process is sensitive to optimizer selection
- Adaptive optimizers often work better
- Learning rate modulation by moving average seems to help stability
- A custom Whetstone-aware optimizer is in early stages

![Optimizers and Learning Rate](chart)
Established Deep Learning Techniques

• The trained neurons can be unreliable
• Redundant output encodings help mitigate this problem
• Similar to ensemble methods
• Reactive neurons feed into softmax during training (for classification)
• During inference, ‘best-matched’ group is used
• On simple datasets, 4-way redundancy is sufficient
Enabling Wide and Easy-to-Implement Adoption

Neuromorphic hardware platforms are appealing for a wide variety of low-power, embedded applications.

Sophistication and expertise required to make use of these platforms creates a high barrier of entry.

Whetstone enables deep learning experts to easily incorporate spiking hardware architectures.
Enabling Wide and Easy-to-Implement Adoption

Networks are portable and hardware-agnostic

Low barrier of entry; built on standard libraries (Keras, Tensorflow, CUDA, etc.)

No post-hoc analysis; no added time complexity

Only simple integrate-and-fire neurons are required

Compatible with standard techniques like dropout and batch normalization
Enabling Wide and Easy-to-Implement Adoption

Neuromorphic Hardware in Practice and Use

Description of the workshop

Abstract – This workshop is designed to explore the current advances, challenges and best practices for working with and implementing algorithms on neuromorphic hardware. Despite growing availability of prominent biologically inspired architectures and corresponding interest, practical guidelines and results are scattered and disparate. This leads to wasted repeated effort and poor exposure of state-of-the-art results. We collect cutting edge results from a variety of application spaces providing both an up-to-date, in-depth discussion for domain experts as well as an accessible starting point for newcomers.

Goals & Objectives

This workshop strives to bring together algorithm and architecture researchers and help facilitate how challenges each face can be overcome for mutual benefit. In particular, by focusing on neuromorphic hardware practice and use, an emphasis on understanding the strengths and weaknesses of these emerging approaches can help to identify and convey the significance of research developments. This overarching goal is intended to be addressed by the following workshop objectives:

- Explore implemented or otherwise real-world usage of neuromorphic hardware platforms
- Help develop ‘best practices’ for developing neuromorphic-ready algorithms and software
- Bridge the gap between hardware design and theoretical algorithms
- Begin to establish formal benchmarks to understand the significance and impact of neuromorphic architectures


Call: https://easychair.org/cfp/nipu2018