Mosaics

PRESENTED BY
Brad Aimone
Spiking neural algorithms
- Hand-crafted circuits of spiking neurons
- Model of parallel computation
- Energy efficiency through event-driven communication and high fan-in logic

Artificial neural networks
- Generic layers of non-linear nodes
- SGD optimization of weights
- Powerful machine learning capabilities through learning sequential non-linear mappings and function approximation

Neuroscience-constrained algorithms
- Circuit architecture based on local and regional neural connectivity
- Computation incorporates broad range of neural plasticity and dynamics
- Generally still unexplored from algorithms perspective
cost* (power, time, space...)

application size

network size, input size, ...

Beyond certain scale, neuromorphic approaches become competitive

CPUs / GPUs

* cost depends on computing work & application requirements
Neural Networks

- Better-suited for neuromorphic hardware than many other machine learning techniques

- ANNs only became broadly world class when they reached substantially large sizes

Matrix Multiplication

- Neural algorithms can improve implementation of Strassen-techniques

- Strassen techniques only make sense for large matrix multiplications

Partial Differential Equations

- Neural algorithms can efficiently implement Monte Carlo solutions for solving diffusion-based PDEs

- Monte Carlo methods make most sense for high-dimensional PDEs
*cost depends on computing work & application requirements

What to do when neuromorphic hardware is smaller than crossover point?

* cost depends on computing work & application requirements

application size
(network size, input size, ...)

CPUs / GPUs

Neuromorphic
cost* (power, time, space...)

Reduce overhead for neuromorphic HW (easier programming, better I/O, etc)

application size (network size, input size, ...)

* cost depends on computing work & application requirements

CPUs / GPUs

Neuromorphic

WHETSTONE

FUGU
*cost depends on computing work & application requirements

---

* cost depends on computing work & application requirements

---

Application size
(network size, input size, ...)

---

Make neuromorphic HW suitable for bigger applications

---

Cost (power, time, space ...)

---

CPUs / GPUs

---

Neuromorphic
Memory is cheap. We should take advantage of that.

More synapses (whether as arrays or tables or whatever) is not really the problem.

Communication is expensive. And it scales poorly with neurons and synapses...

Can we formulate an approach to mitigate this?

Neurons are not as cheap. And we need a lot of them.

But depending on the algorithm, they are often unused for large periods of time.

Can we better take advantage of what we have?
Obviously this won’t work for everything

But it will work for a lot of potential applications
Machine Learning

Whetstone

Convolutions

k-Nearest Neighbor

Support Vector Machines

Scientific Computing Application
Fugu = pufferfish (why? Pufferfish have spikes...)

Machine Learning Applications
- Whetstone (Spiking for Keras)
- Specialized Deep Learning Extensions

Numerical Computing Applications
- C++ API
- Various Spiking Kernels (xCorr, SpikeSort, Strassen)
- Neural Random Walkers
- Custom Corelets
- N2A

Spiking Neuromorphic Platforms
Thanks everyone for coming to NICE 2019!