SAND2025-01773C

A Probabilistic Future for Neuromorphic Computing

Brad Aimone

Center for Computing Research

Sandia National Laboratories

There is not a single roadmap for neuromorphic computing

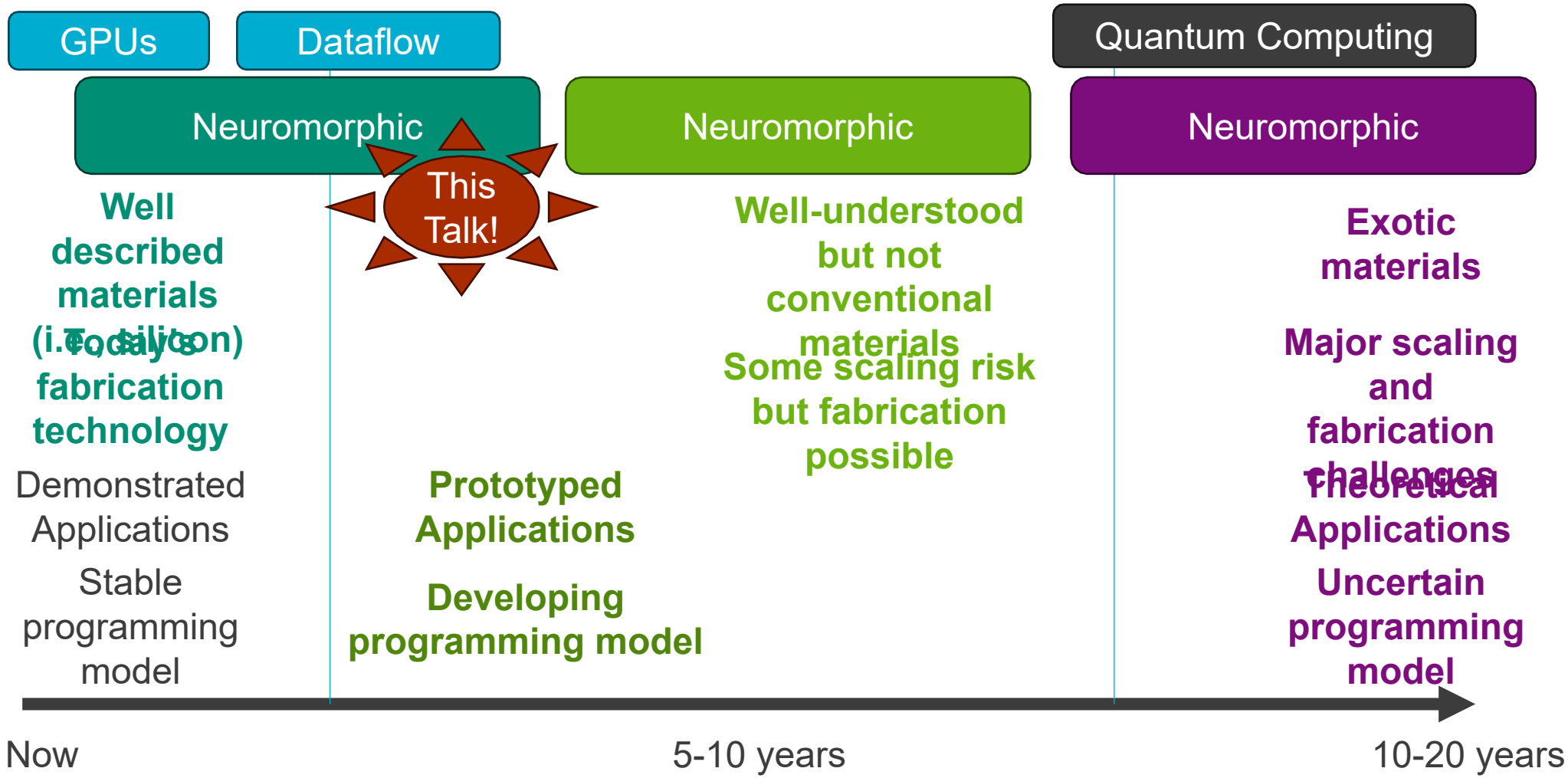
- Active research across the technology stack
  - New post-CMOS materials (memristors, ECRAM, MTJs, quantum materials, …)
  - Non-digital devices (analog, stochastic, optical, …)
  - Bio-inspired circuits (reconfigurable, dendrites, learning, …)
  - Neuromorphic architectures (spiking, event-driven sensors, ANN accelerators, …)
  - Software paradigms (compilers, intermediate representations, …)
  - Neuromorphic algorithms

When is it ready for prime time?
What is needed to get there?

When will neuromorphic computing be a reality?

GPUs	Dataflow		Quantum Computing
Well described materials (i.e., silicon)		Well-understood but not conventional materials	Exotic materials
Today's fabrication technology		Some scaling risk but fabrication possible	Major scaling and fabrication challenges
Demonstrated Applications		Prototyped Applications	Theoretical Applications
Stable programming model		Developing programming model	Uncertain programming model
Now		5-10 years	10-20 years

Neuromorphic computing has promise at different time scales

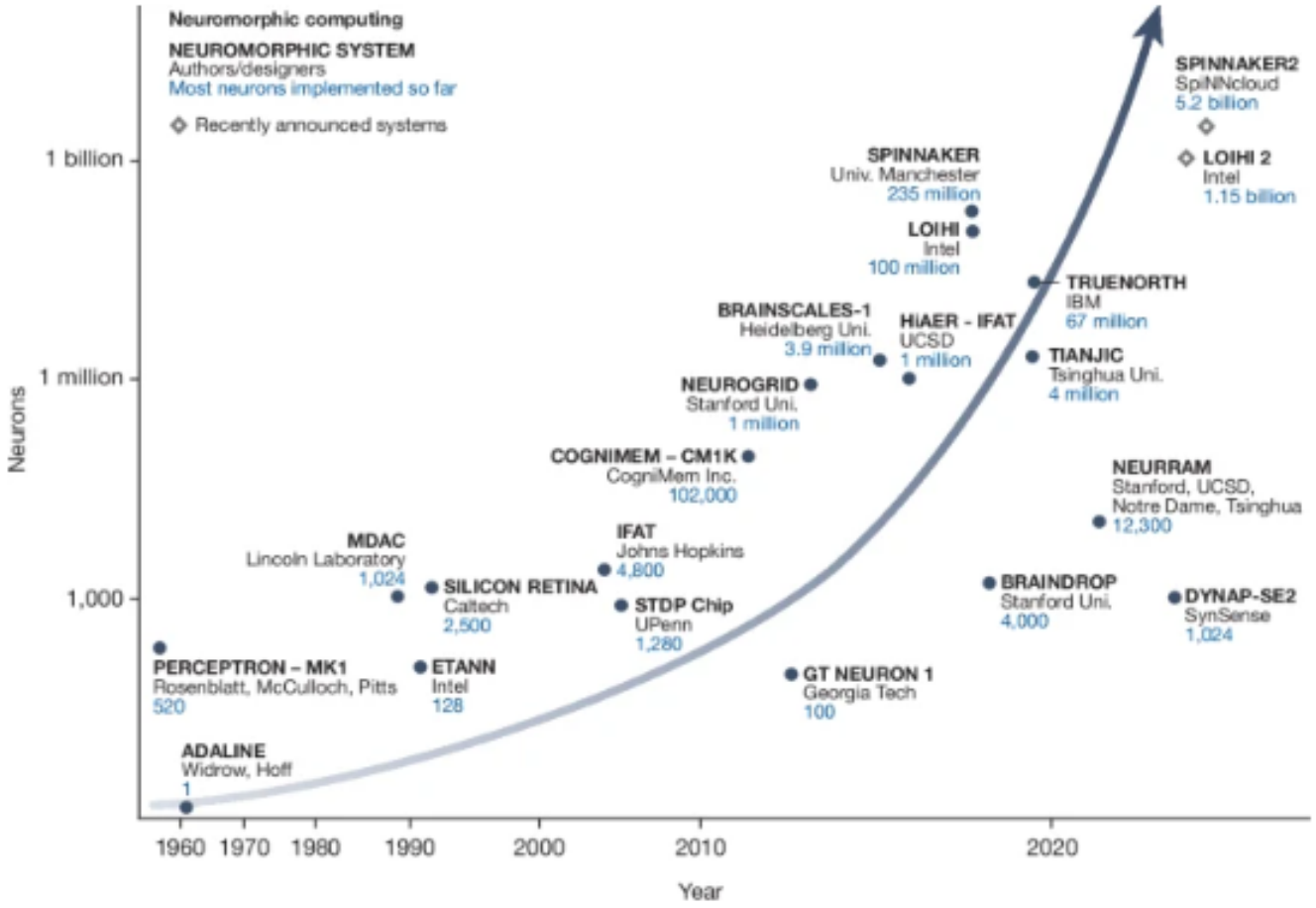


Today's digital neuromorphic systems are approaching brain-scale

- Systems like Intel's Loihi 2 and SpiNNCloud's SpiNNaker 2 can surpass 1 billion neurons
- Individual chips are ~1 million neurons and ~1 Watt
- Fully CMOS (little fabrication risk)
- Digital or Digital + Analog hybrid

- Future devices and novel materials can amplify potential impact

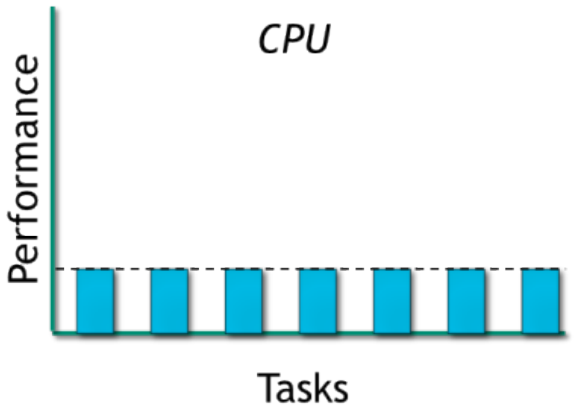
Fig. 1: Progression of neuromorphic computing systems.



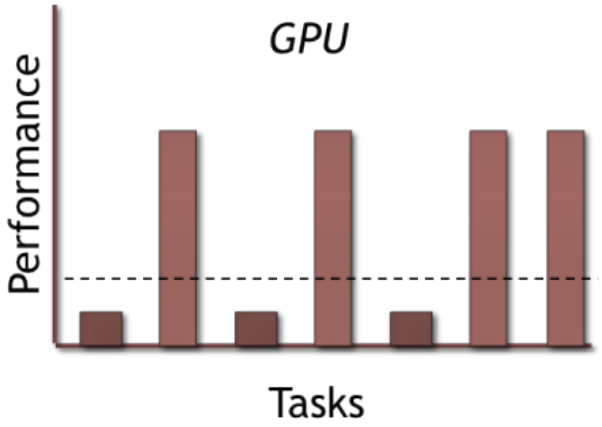
Kudithipudi et al., Nature 2025

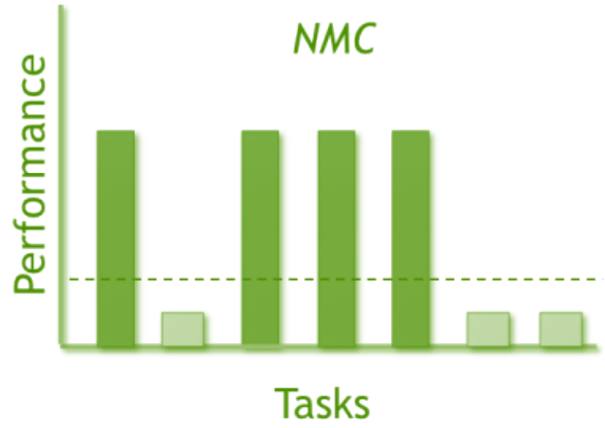
Neuromorphic is likely similar to GPUs in degree of specialization

Truly General Purpose

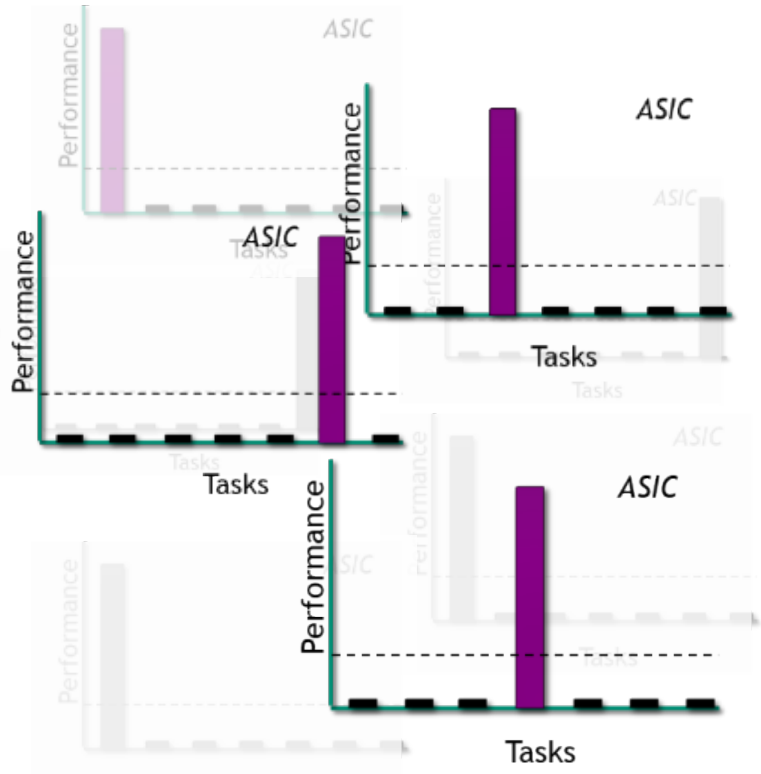


Specialized General Purpose

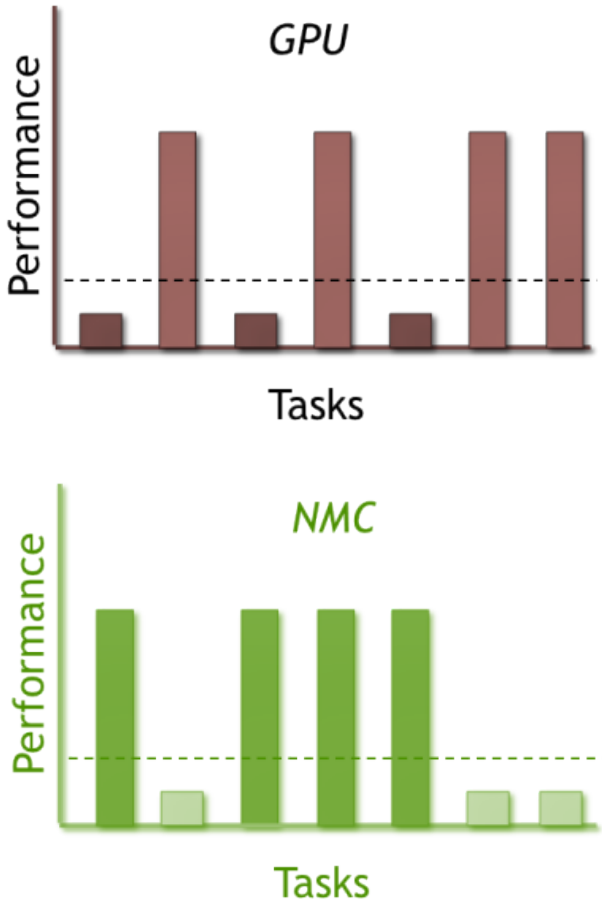




Application Specific



Claim: the major challenge to neuromorphic computing today is the algorithm impact



Identifying neuromorphic advantages today will

- Communicate the fundamental value proposition of neuromorphic specific
- Compare the scope of neuromorphic impact to alternatives (GPUs, accelerators, etc)
- Clarify what aspects of today's neuromorphic architectures need to be improved
- Justify cost of moving to new non-CMOS materials

So is there actually a neuromorphic advantage?

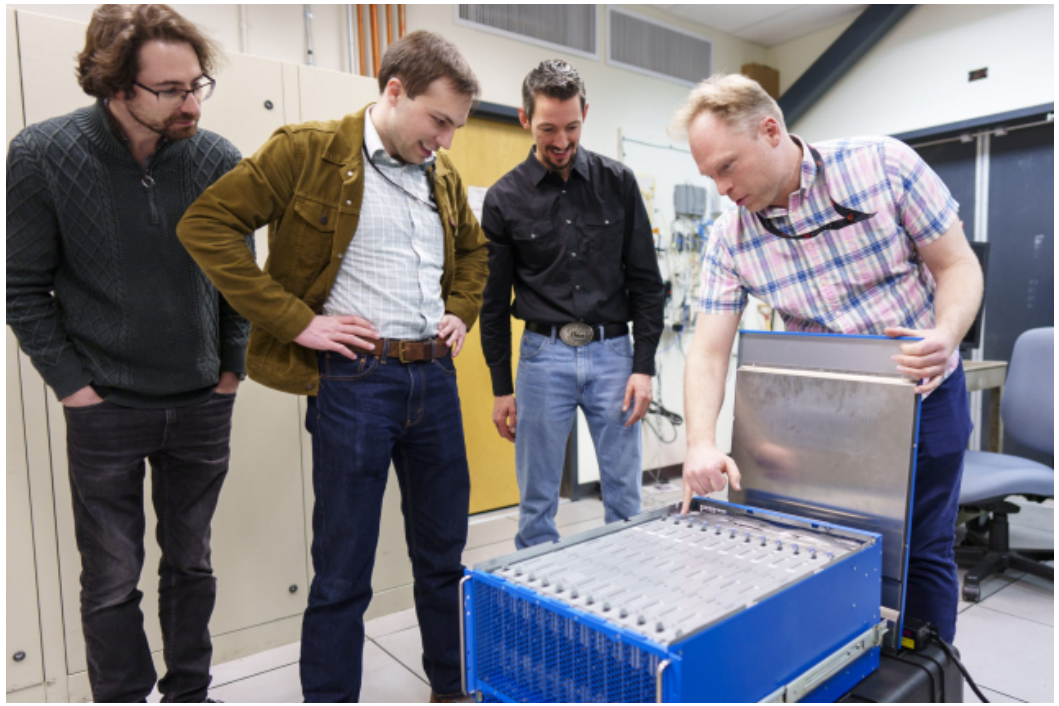
Neuromorphic advantage: an algorithm that surpasses conventional architectures on one performance axis (energy, time, accuracy) while demonstrating equivalent (or better) performance on all other axes

- Machine Learning 
- Scientific Computing 
- Sensing and Autonomy 

What is neuromorphic computing today?

In digital silicon (CMOS) technology
- Over 1 billion neuron system at Sandia
  - Roughly the number in a parrot or small primate brain
- Neurons are “simulated” in an efficient way
- Generally a leaky integrate-and-fire model

Analog systems
- Wide range of technologies, but far smaller
- Emulate the brain’s biophysics in different materials
- Wide range of neural dynamics emulated

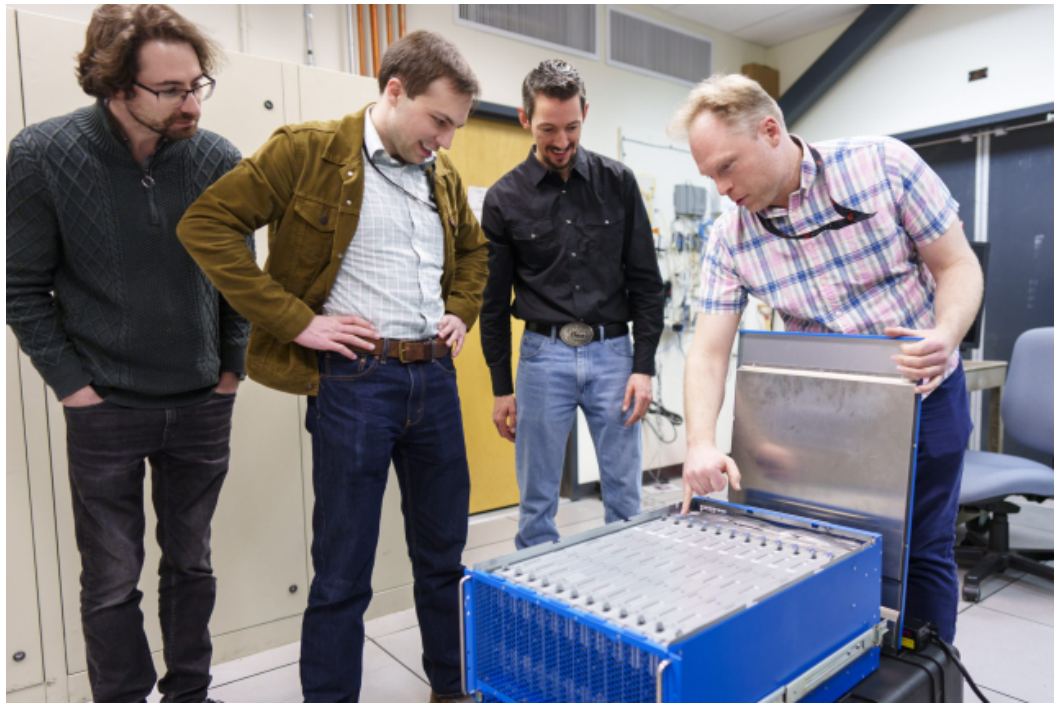


Left to right: William Chapman, Brad Theilman, Craig Vineyard, Mark Plagge

What is neuromorphic computing today?

In digital silicon (CMOS) technology
- Over 1 billion neuron system at Sandia
  - Roughly the number in a parrot or small primate brain
- Neurons are “simulated” in an efficient way
- Generally a leaky integrate-and-fire model

Analog systems
- Wide range of technologies, but far smaller
- Emulate the brain’s biophysics in different materials
- Wide range of neural dynamics emulated



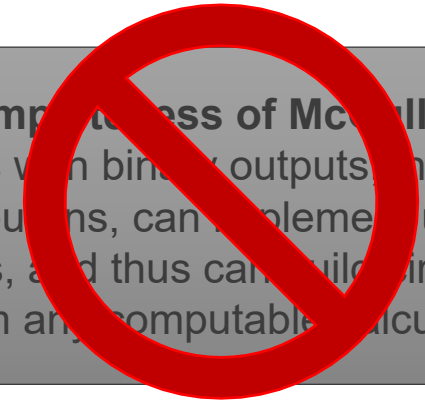
Left to right: William Chapman, Brad Theilman, Craig Vineyard, Mark Plagge

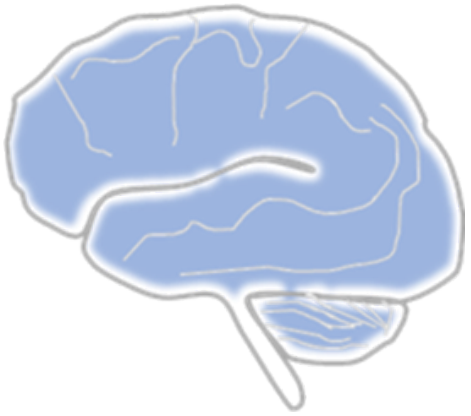
Rationale: Efficiency protects us from being trapped by two trivial conclusions

Universal Function Approximation
For any continuous f, there is a feedforward neural network Φ that approximates f

No guarantee of efficiency

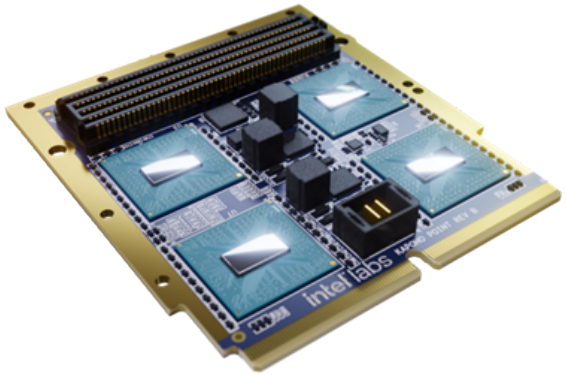
Turing Completeness of McCulloch-Pitts
Neurons with binary outputs, including spiking neurons, can implement universal logic gates, and thus can build circuits that perform any computable calculation



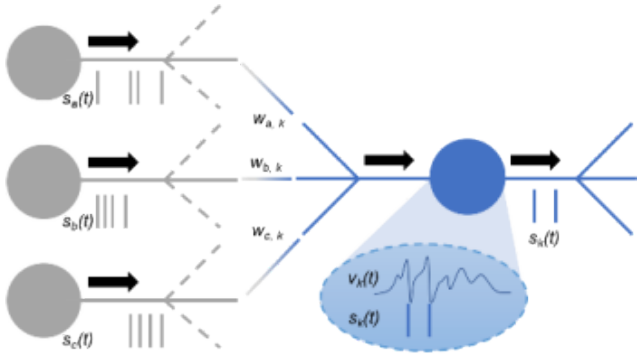


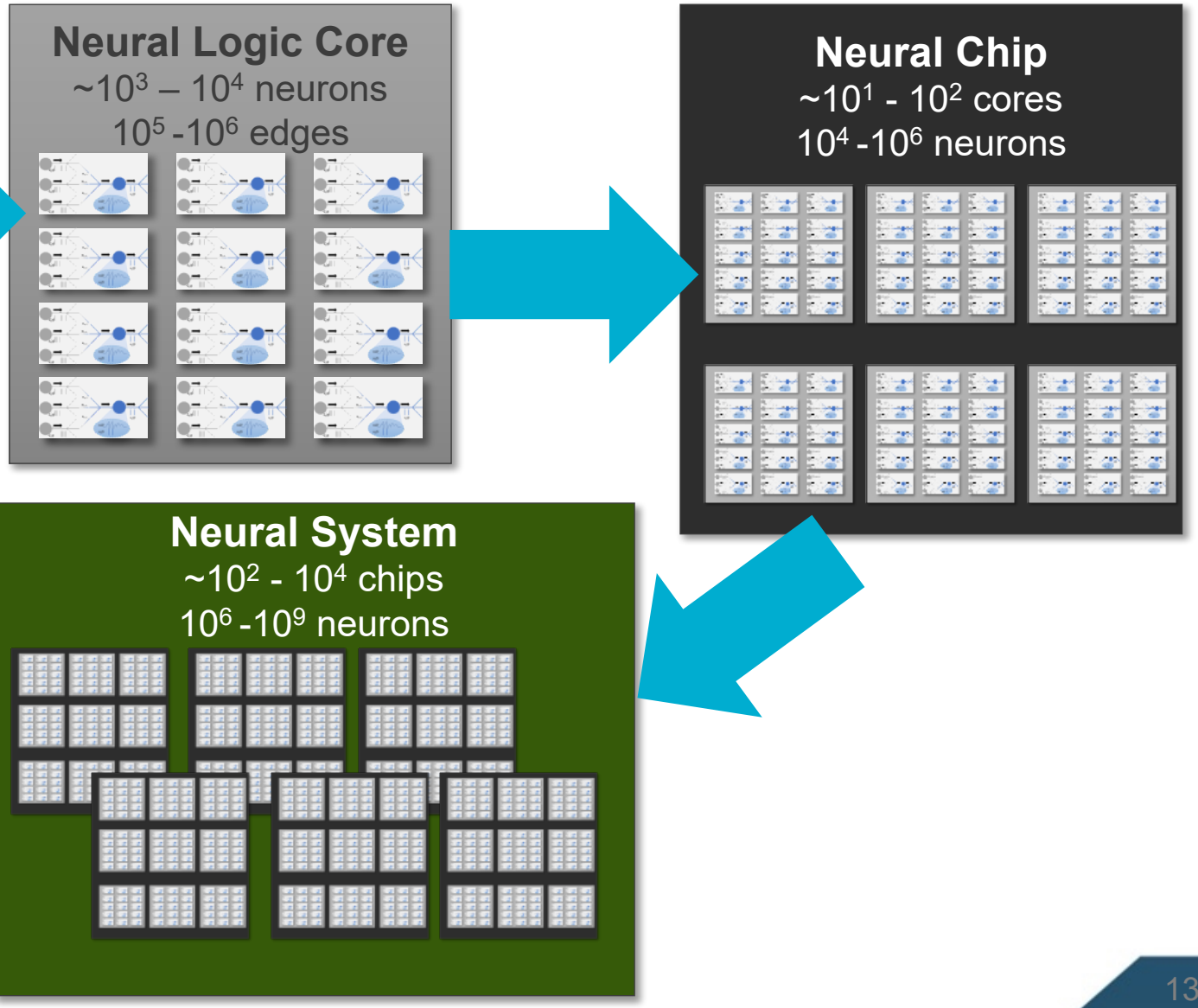
The brain (very) likely only computes what the brain is efficient at computing

Neuromorphic should have many of the same restrictions



Spiking neuromorphic today: Overview





Computational Primitives:

Spiking Neurons (vertices/nodes)

Synapses (connections/edges)

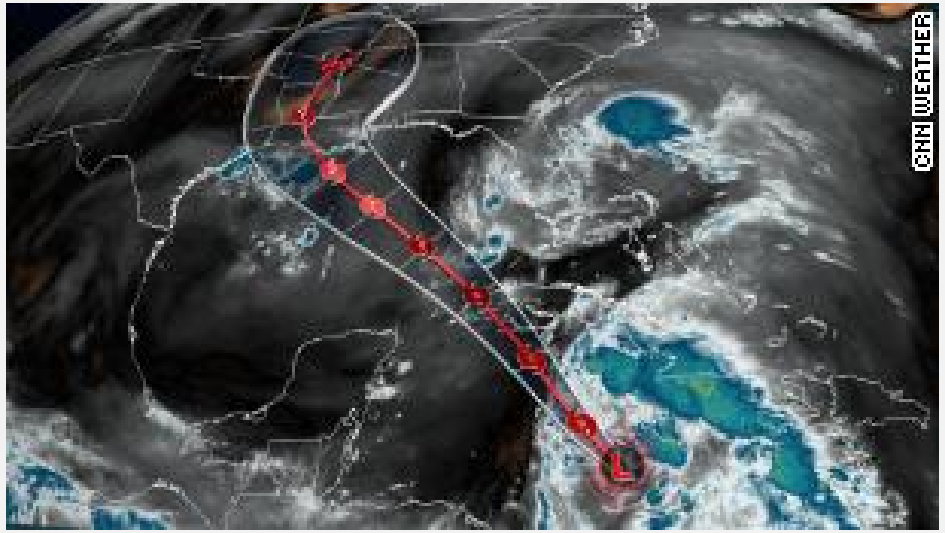
Programmable as arbitrary graphs

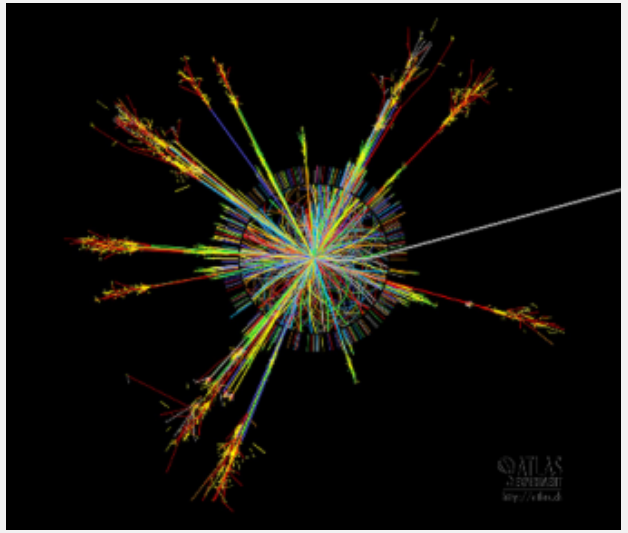
- Edges: Directed and weighted
- Nodes: Threshold gate logic + time
- *Artificial neural networks are a special case*
- Programmability, theoretical, analysis, and software are open research questions

Computing applications face challenges in uncertainty

Artificial Intelligence

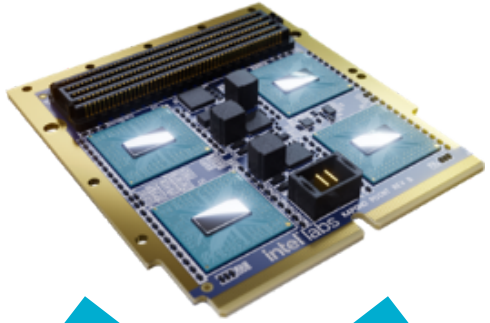
- Bayesian neural networks are appealing yet often computationally intractable



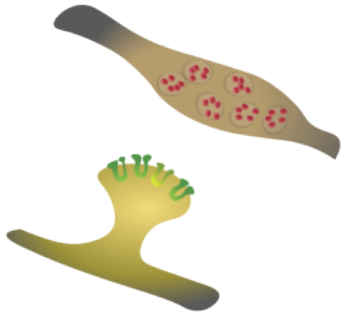


Modeling and Simulation

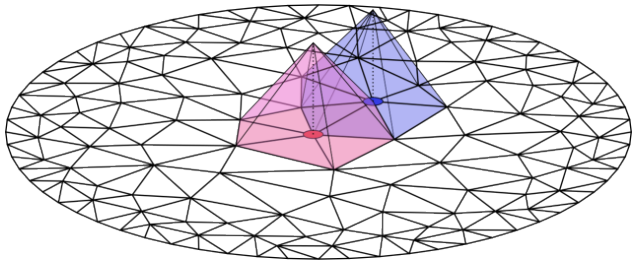
- Modeling uncertainties is critical in the use of even fully deterministic simulations
- Many applications are inherently stochastic in their physics and are best modeled using probabilistic methods



Problems that require widespread sampling



Problems that are naturally spatially distributed



Diffusion

Diffusion can be modeled either as a deterministic PDE or a stochastic process

- For an initial distribution of particles, P_0, what is distribution of particles at time *t?*
- PDE solves the following equation

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t}{\partial x^2}$$

- Stochastic process implements many random walkers to statistically approximate a solution
  - Mean position of *N* walkers approaches expected mean of deterministic solution at rate of 1/sqrt(N)

“Naïve” way to run diffusion on GPUs

```
void rw_mix(curandState_t* states, int n, int *x, int *y)
{
    int i = blockIdx.x*blockDim.x+threadIdx.x;
    int rand_1;
    int rand_2;
    if (i < n){
        for (int t=0; t<100000; t++){
            rand_1 = curand(&states[i]) % 2;
            rand_2 = curand(&states[i]) % 2;
            if (rand_1 == 1){
                if(rand_2 ==1) x[i]++;
                else x[i]-=1;
                }
            else{
                if (rand_2 == 1) y[i]++;
                else y[i]-=1;
            }
        }
        y[i] = y[i]%21;
        x[i] = x[i]%21;
    }
}
```

For every particle “owned” by this GPU

… loop through 100,000 time steps

… draw two random numbers

50% chance to pick up or down…

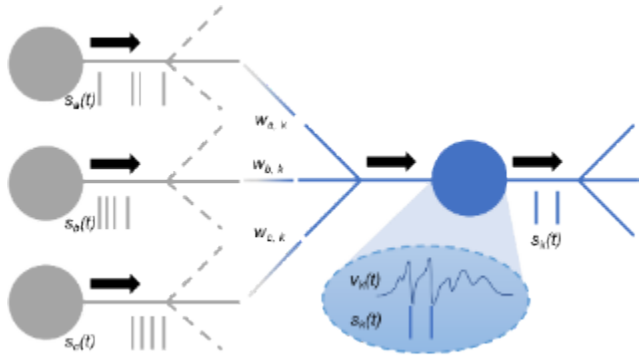
50% chance to pick left or right…

… mod back to torus

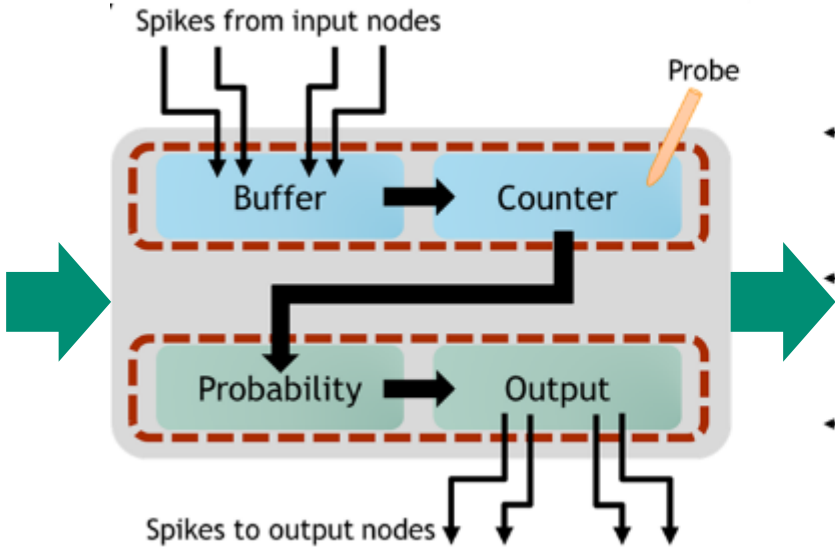
- ✓ Easy to do
- ✓ Embarassingly parallel over particles
- ✓ Mostly independent of size of state space

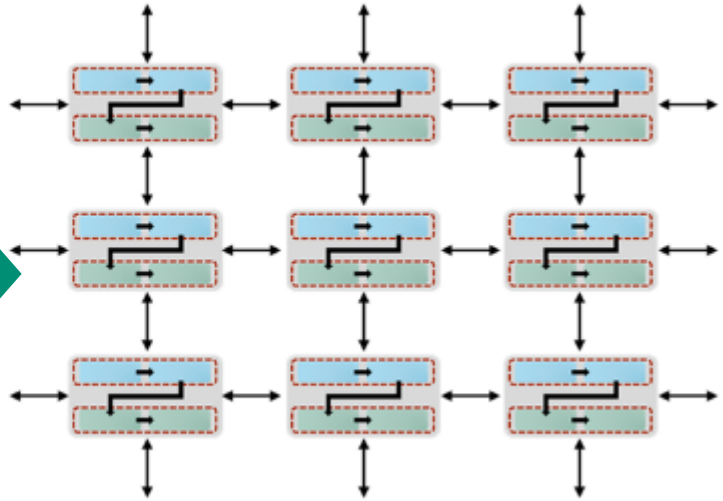
- X Particle location in computer is independent from location in model
- X High cost of inhomogeneity
- X Lots of random number draws

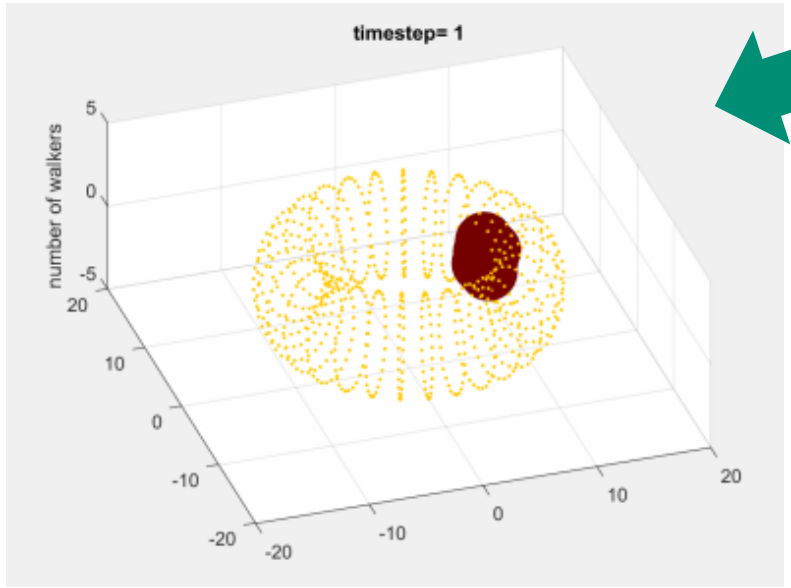
Flip the approach! Use neurons to represent *state space* of Monte Carlo and use spikes to represent particles

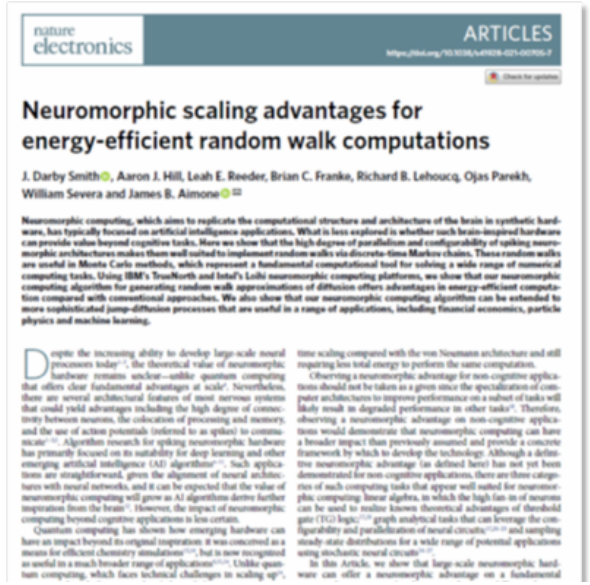


Leaky integrate-and-fire neuron



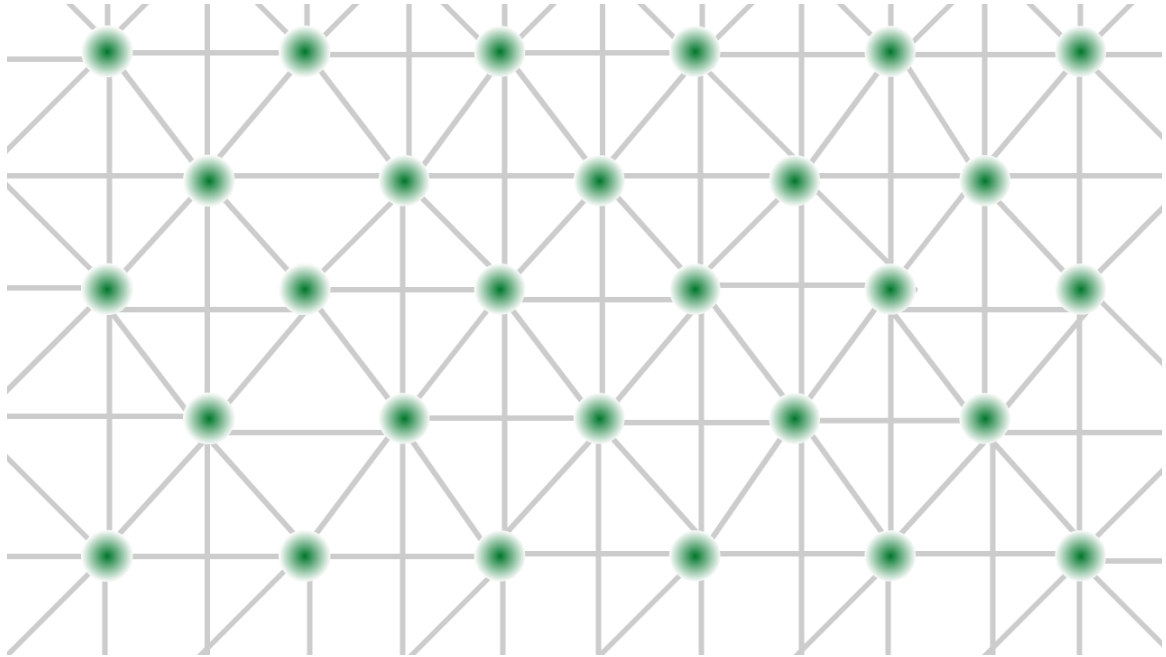






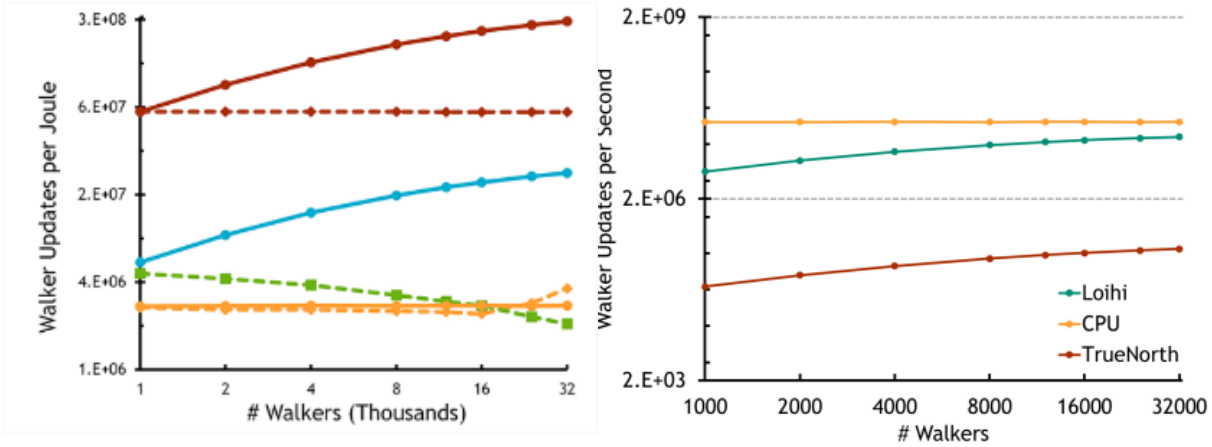
Smith et al., Nature Electronics 2022

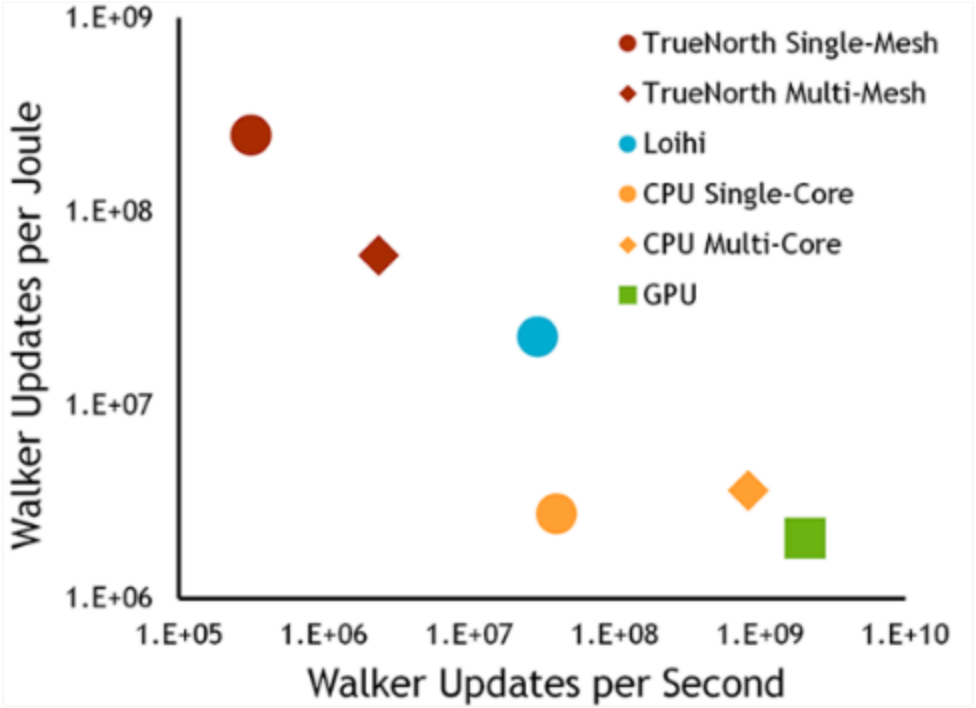
Neuromorphic computing advantage appears to be when an algorithm can split the task across computational graph with sparse communication



We can identify a neuromorphic advantage for simulating random walks

We define a *neuromorphic advantage* as an algorithm that shows a demonstrable **advantage** in terms of one resource (e.g., energy) while exhibiting comparable **scaling** in other resources (e.g., time).





Smith et al., Nature Electronics 2022

Math: What PDEs can these stochastic processes be useful for?

Class of Partial Integro-Differential Equations:

$$\frac{\partial}{\partial t}u(t,\boldsymbol{x}) = \frac{1}{2}\sum_{i,j}(\boldsymbol{a}\boldsymbol{a}^{\top})_{i,j}(t,\boldsymbol{x})\frac{\partial^2}{\partial x_i \partial x_j}u(t,\boldsymbol{x}) + \sum_{i} b_i(t,\boldsymbol{x})\frac{\partial}{\partial x_i}u(t,\boldsymbol{x})$$

$$+\lambda(t,\boldsymbol{x}) \quad u\ t,\boldsymbol{x}+\boldsymbol{h}(t,\boldsymbol{x},q)\ - u(t,\boldsymbol{x})\ \phi_Q(q;t,\boldsymbol{x})\mathrm{d}q$$

$$+c(t,\boldsymbol{x})u(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), \qquad x \in \mathbb{R}^d, t \in [0,\infty).$$

Stochastic Process:

NMC Hardware Simulates This Stochastic Process

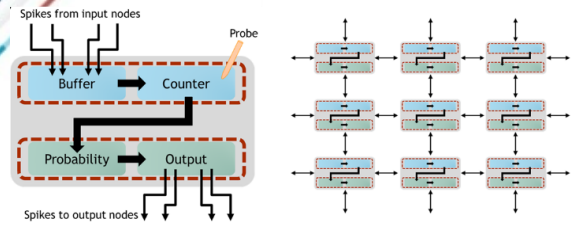
$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{b}\ t,\boldsymbol{X}(t)\ \mathrm{d}t + \boldsymbol{a}\ t,\boldsymbol{X}(t)\ \mathrm{d}\boldsymbol{W}(t) + \boldsymbol{h}(t,\boldsymbol{X}(t),q)\mathrm{d}P\ t;Q,\boldsymbol{X}(t)\ .$$

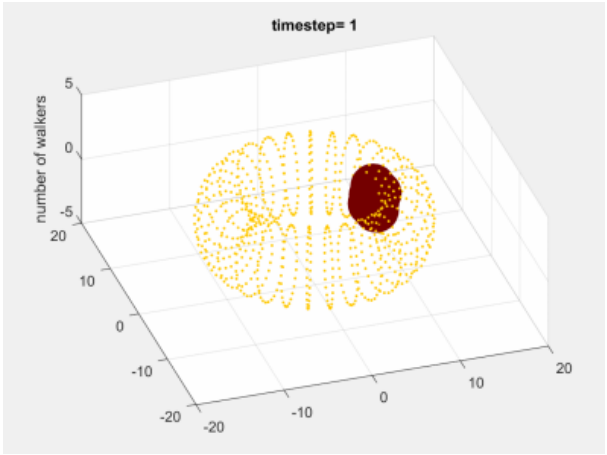
Monte Carlo Approximates This Expectation

Solution to initial value problem (u(0,x)=g(...

$$u(t,\boldsymbol{x}) = \mathbb{E}\ g\ \boldsymbol{X}(t)\ \exp\int_0^t c\ s,\boldsymbol{X}(s)\ \mathrm{d}s\ + \int_0^t f\ s,\boldsymbol{X}(s)\ \exp\int_0^s c\ \ell,\boldsymbol{X}(\ell)\ \mathrm{d}\ell\ \mathrm{d}s\ \ \boldsymbol{X}(0)=\boldsymbol{x}\ .$$

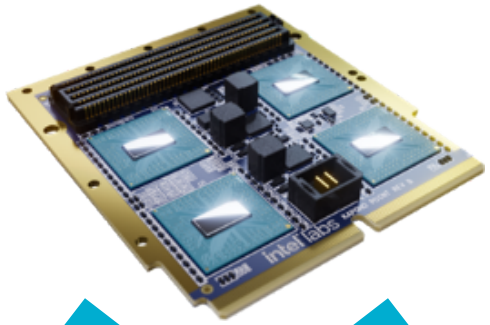
Smith et al., Nature Electronics 2022

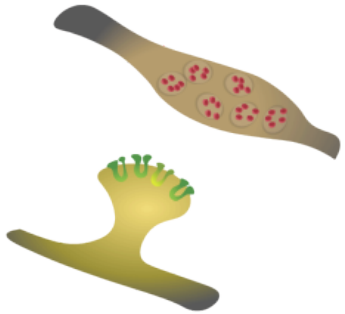




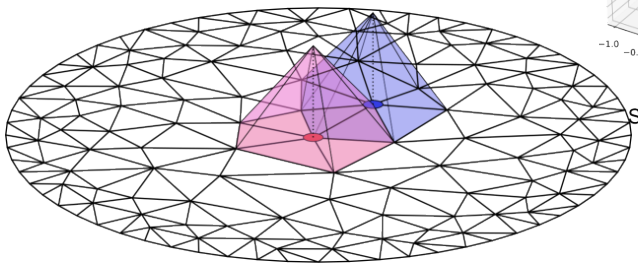
Smith et al., Nature Electronics, 2022

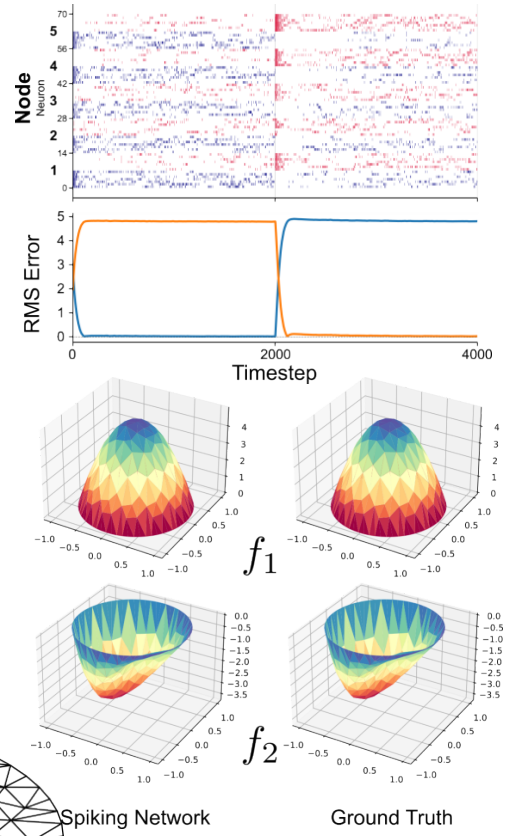
Neuromorphic hardware can *efficiently* solve stochastic Monte Carlo simulations





Neuromorphic hardware can *efficiently* solve finite element method simulations





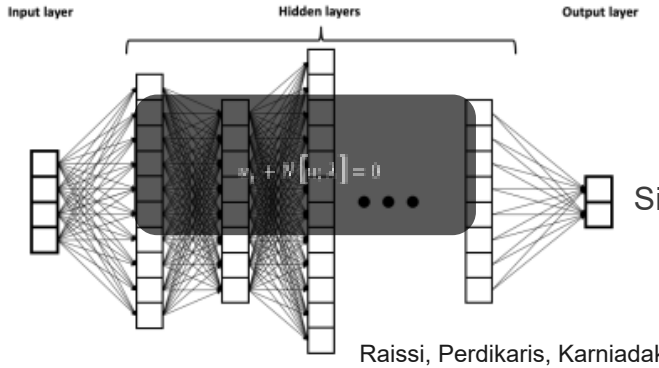
Theilman et al., in review

Solving partial differential equations is a major challenge to emerging computing platforms

$$\nabla^2 u = f$$

Scientific Machine Learning

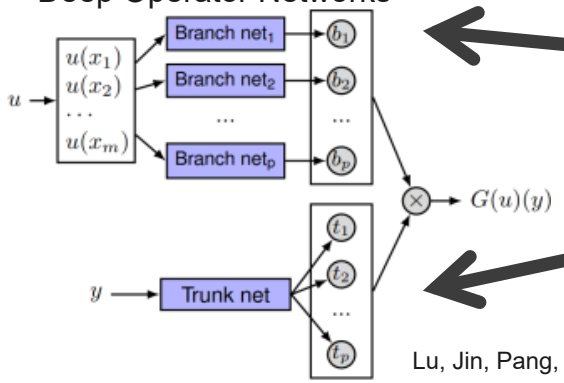
Physics-Informed Neural Networks



Single solution

Raissi, Perdikaris, Karniadakis 2019

Deep Operator Networks

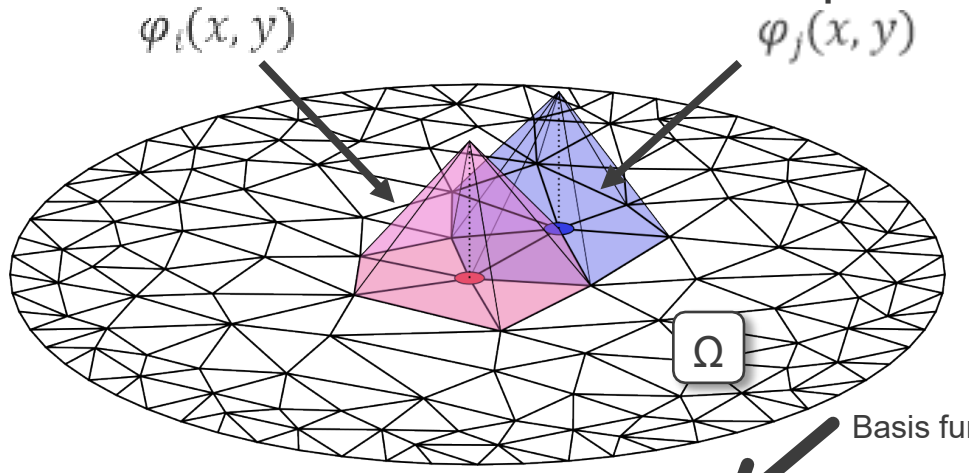


Coefficients

Basis functions

Lu, Jin, Pang, Zhang, Karniadakis 2021

Finite Element Methods – Gold Standard but expensive



Basis functions

$$u(x, y) \approx \sum_i u_i \varphi_i(x, y)$$

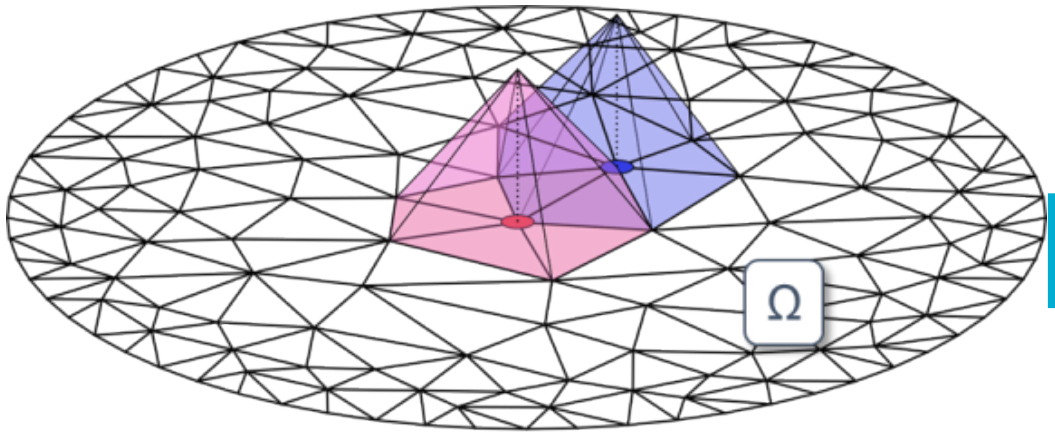
Coefficients

Sparse, linear system

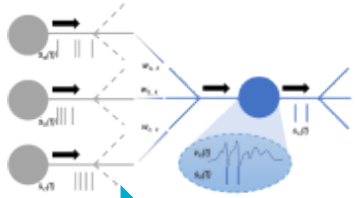
$$A\vec{u} = \vec{b}$$

NEUROMORPHIC **FINITE ELEMENT METHODS?**

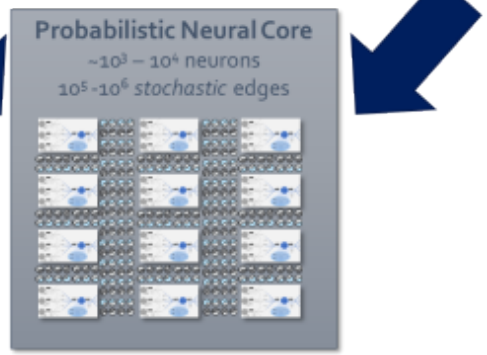
Can we tackle FEM with probabilistic neural hardware?

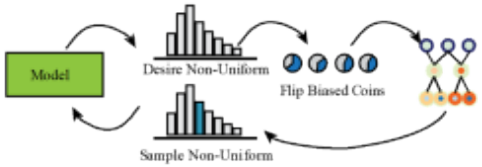


$$u(x,y) \approx \sum_i u_i \varphi_i(x,y)$$

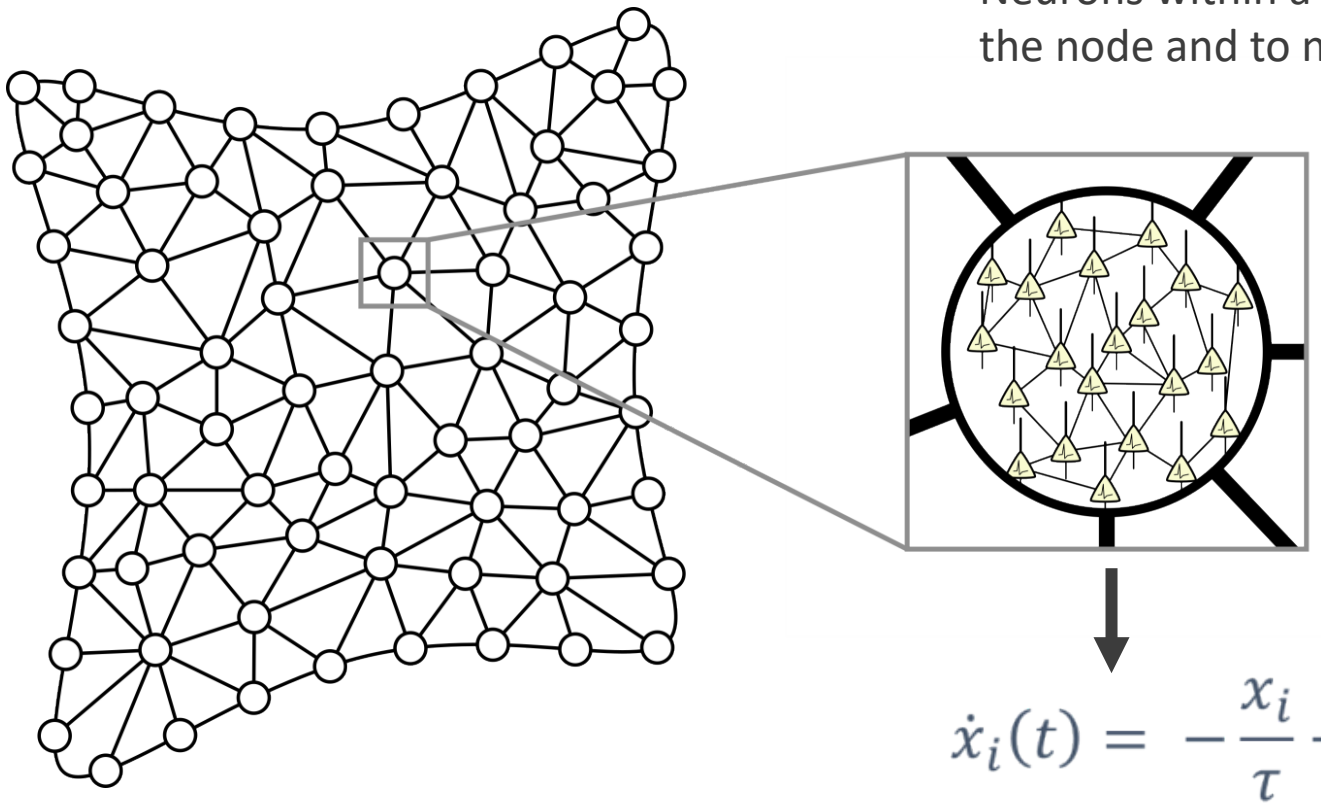


?????





FEM – SNN MAPPING

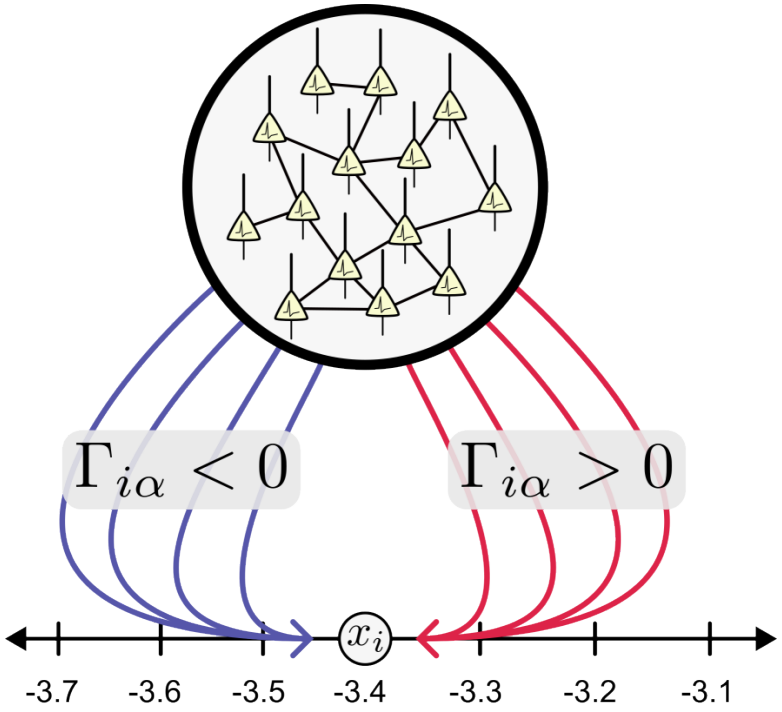


Each node “contains” a population of neurons

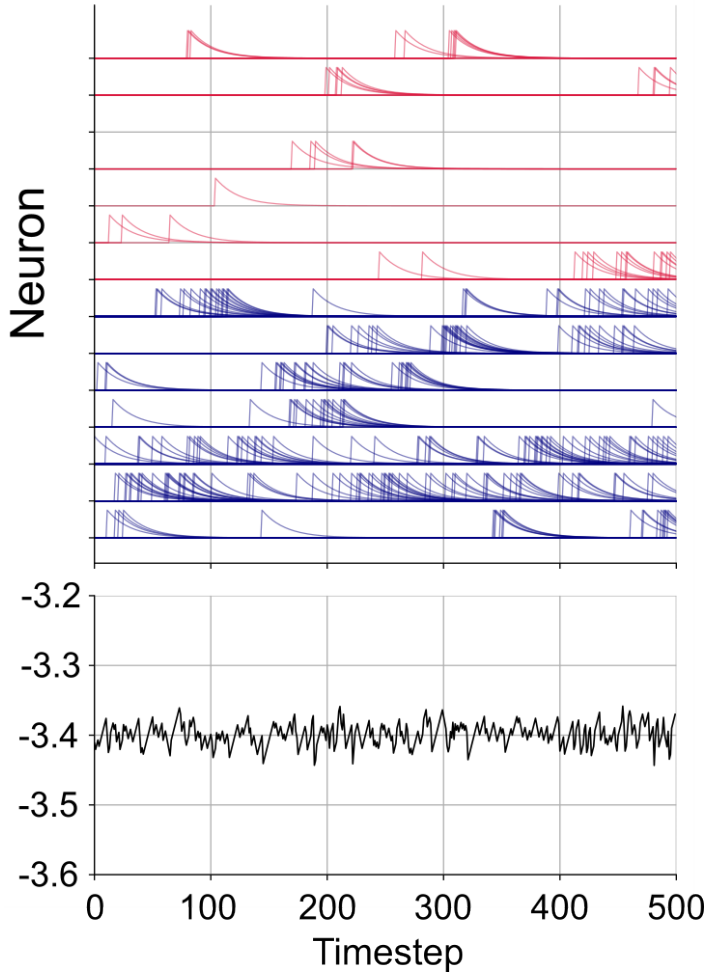
Neurons within a node connect locally within the node and to neighboring nodes

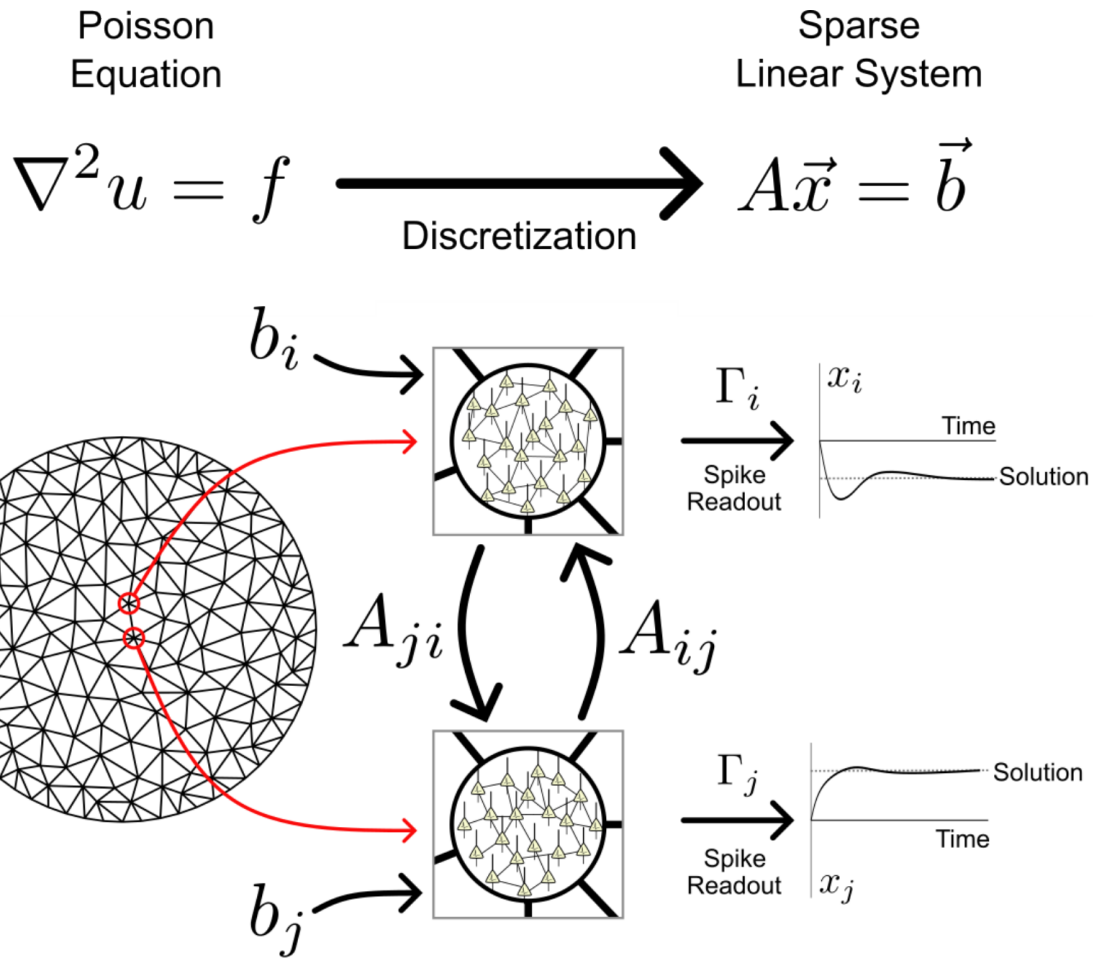
$$\dot{x}_i(t) = -\frac{x_i}{\tau} + \vec{\Gamma}_{in} s_n$$

Neurons within a node project to the solution variable associated to that node



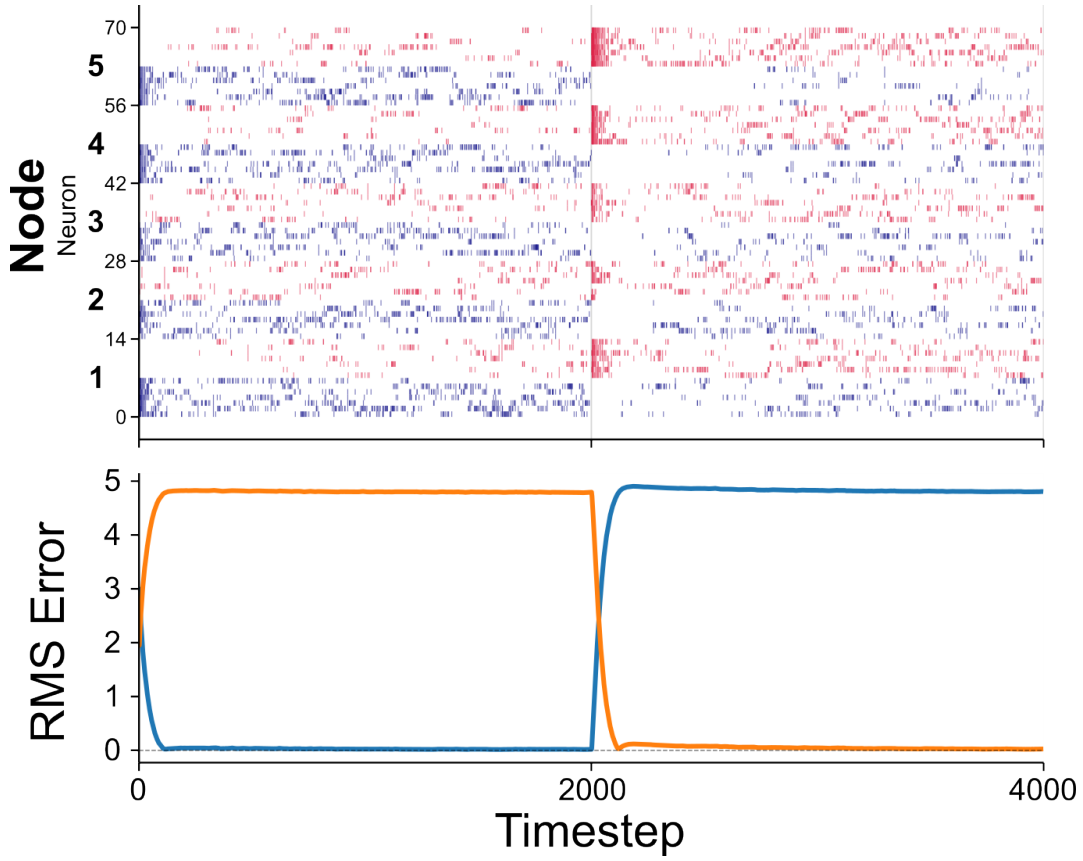
$$\frac{dx_i}{dt} = -\lambda_d x_i + \Gamma_{i\alpha} s_\alpha$$

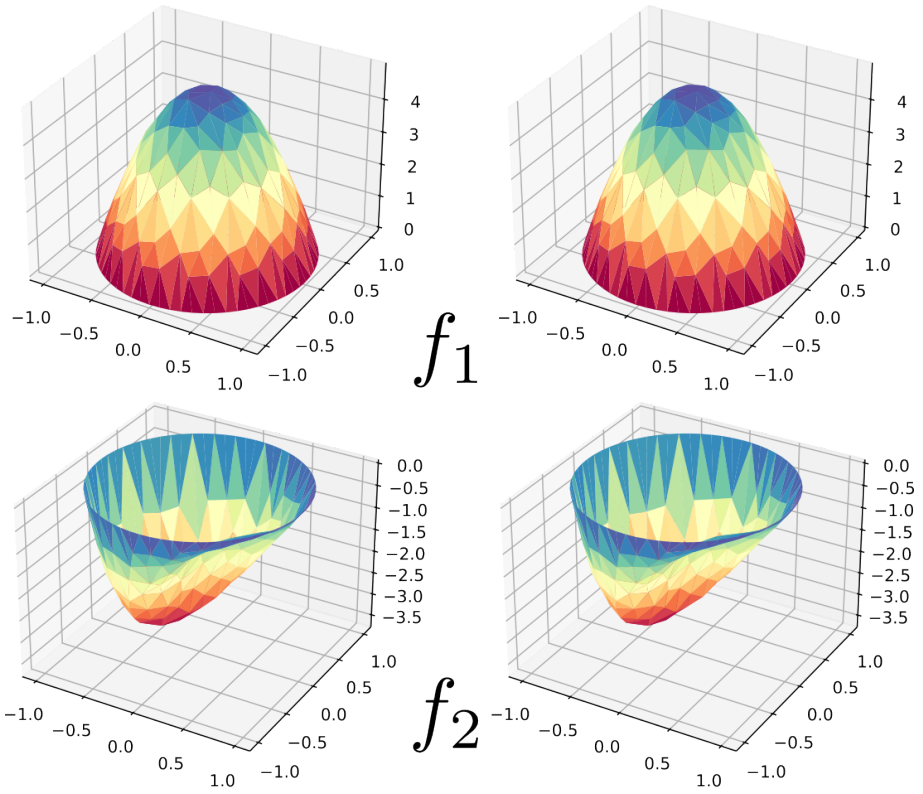




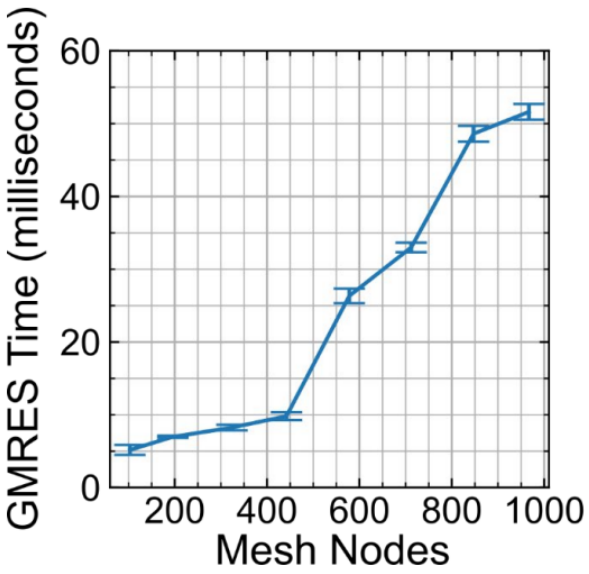
Neuromorphic virtues:

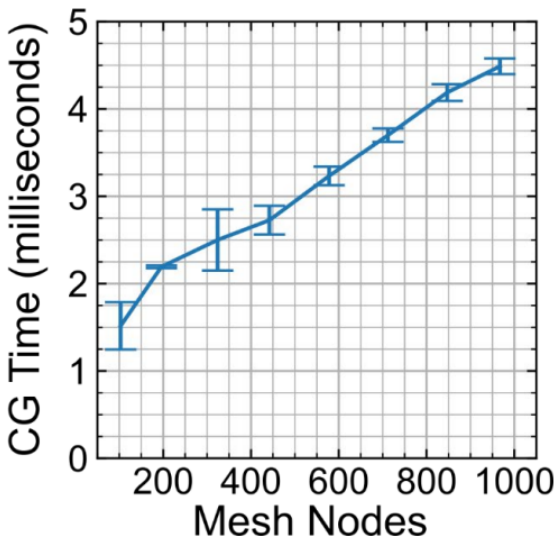
- Locally dense, globally sparse connectivity
- Scalable: neighbors ~ O(1)
- Sparse spiking activity


Node
Neuron
70
56
42
28
14
0
5
4
3
2
1
RMS Error
5
4
3
2
1
0
0
2000
4000
Timestep


f_1
f_2
Spiking Network
Ground Truth

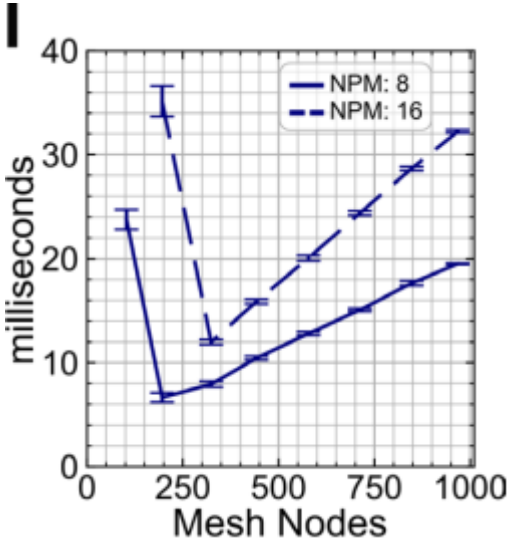
NeuroFEM is similar in speed to GMRES and Conjugate Gradient

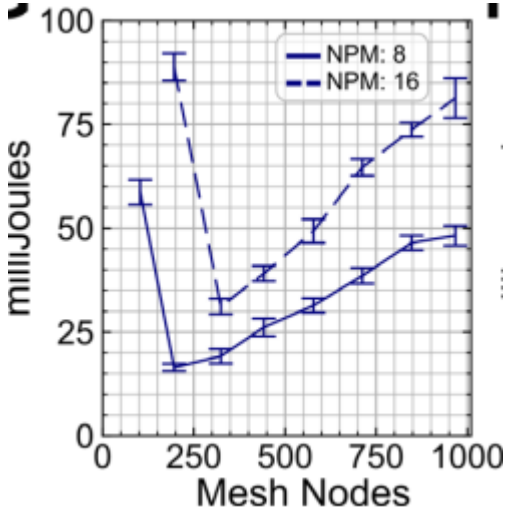




Conjugate gradient is faster at small scales, but can only be used for symmetric matrices

Neuromorphic solution is general

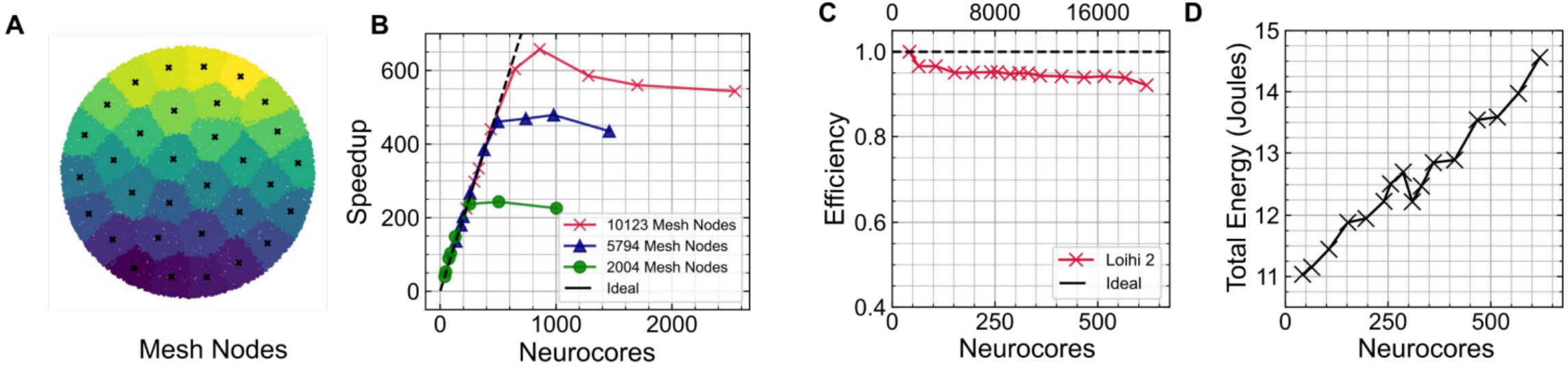




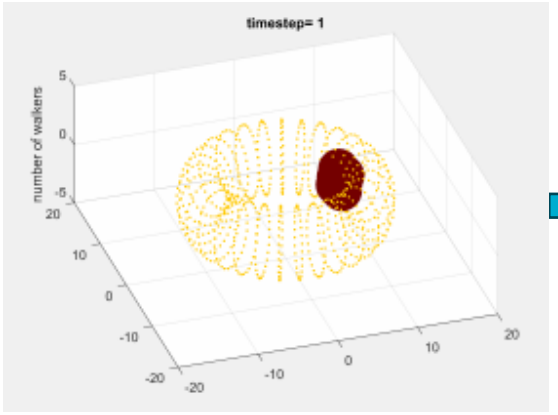
NeuroFEM shows compelling strong and weak scaling

Strong Scaling

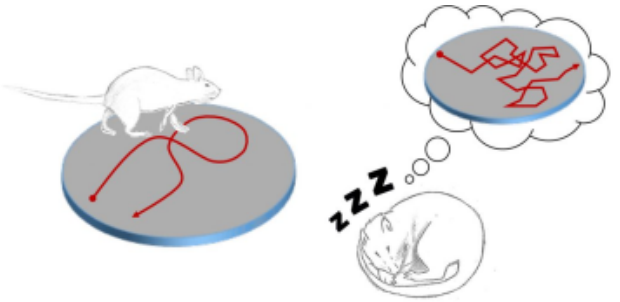
Weak Scaling

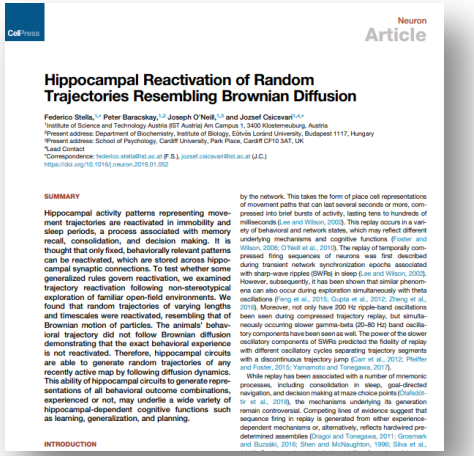


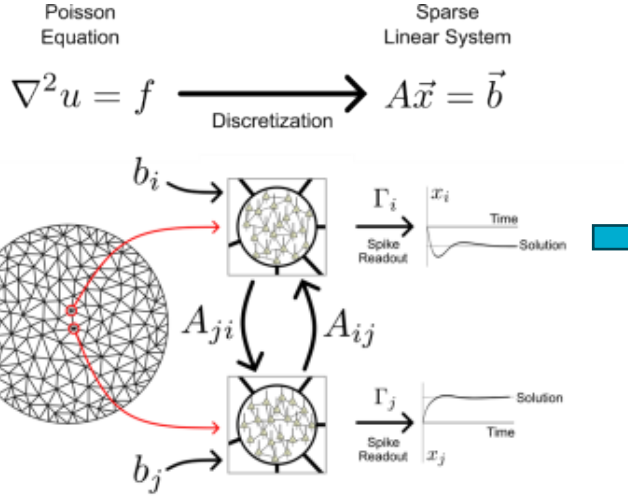
These neuromorphic advantages are perhaps closer to biology than people may realize



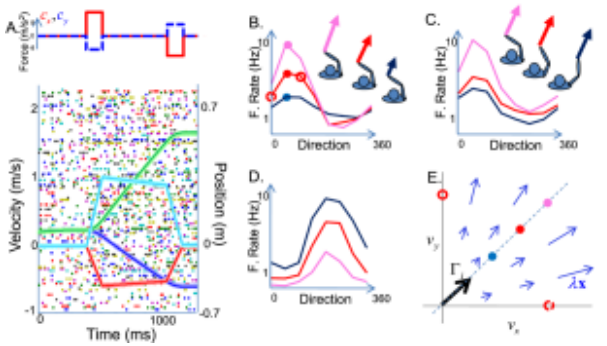
Widespread sampling by neural circuits

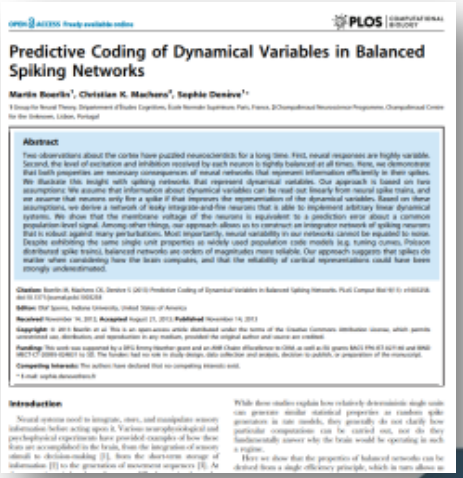






Distributed control dynamics to describe action space





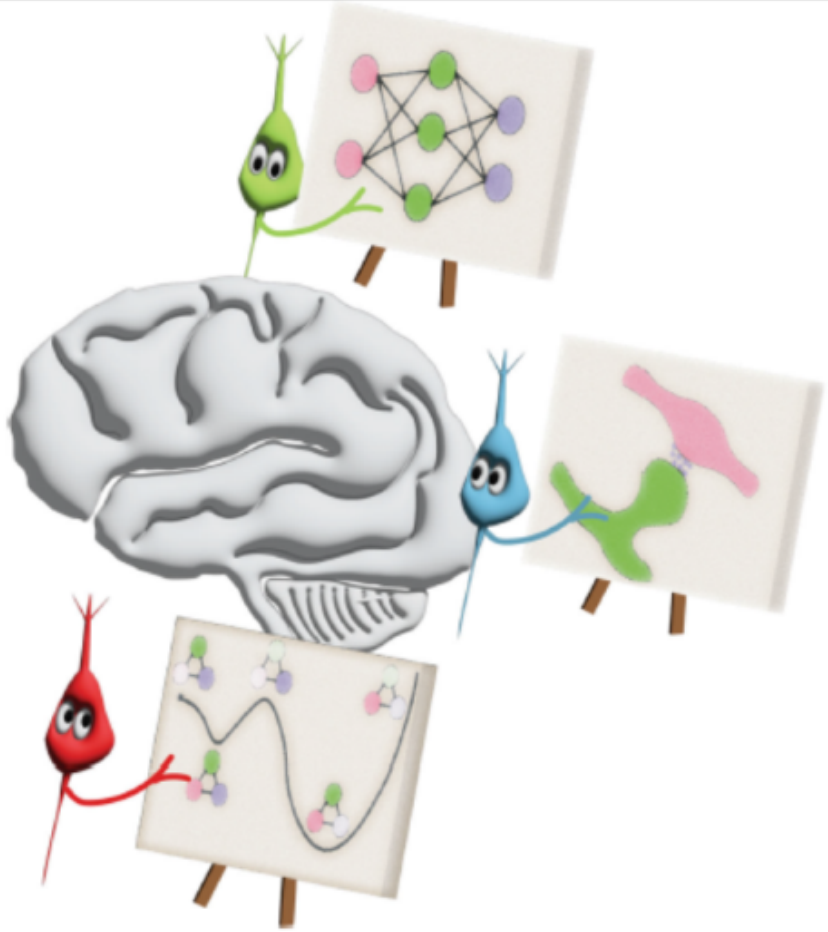
Thank You!

Neuromorphic Monte Carlo Team
- Darby Smith, Michael Krygier, William Severa, Ojas Parekh, Rich Lehoucq, Aaron Hill

Neuromorphic Testbed Team
- Craig Vineyard, Suma Cardwell, Aaron Hill, Srideep Musuvathy, Felix Wang, Fred Rothganger, Frances Chance, Corinne Teeter, Mark Plagge, Ryan Dellana

Neuromorphic FEM
- Brad Theilman

COINFLIPS team
- Shashank Misra, Conrad James, Darby Smith, Suma Cardwell, Brad Theilman, Ojas Parekh, Yipu Wang, Chris Allemang, William Severa, Prasanna Date, Andy Kent, Laura Reim, Les Bland, Bernd Surrow, Jean Anne Incorvia, Jaesuk Kwon, Sam Liu, Katie Schuman, Karan Patel



jbaimon@sandia.gov