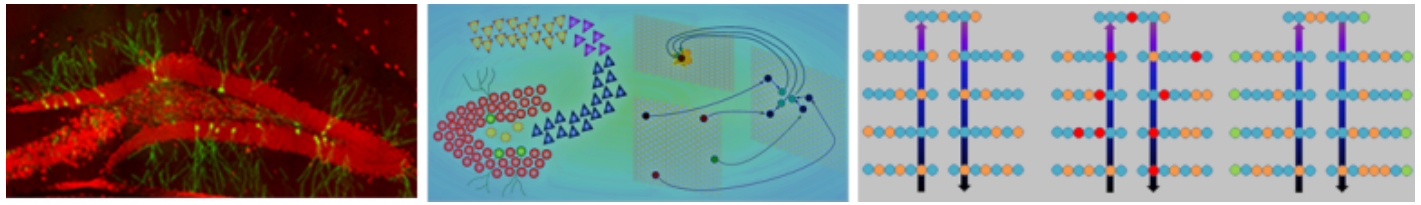
Sandia National Laboratories

SAND2021-3680PE

Neural-Inspired Computation for Scientific Computing



PRESENTED BY

J. Darby Smith

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.

Outline

- High Performance Computing and Energy
- Neural-Inspired Computing
- Neural Random Walks and Applications
- Learning Random Walks – An Inverse Problem
- Non-Academic Career Paths in Mathematics and Computer Science
  - Sandia's Neural Exploration & Research Lab

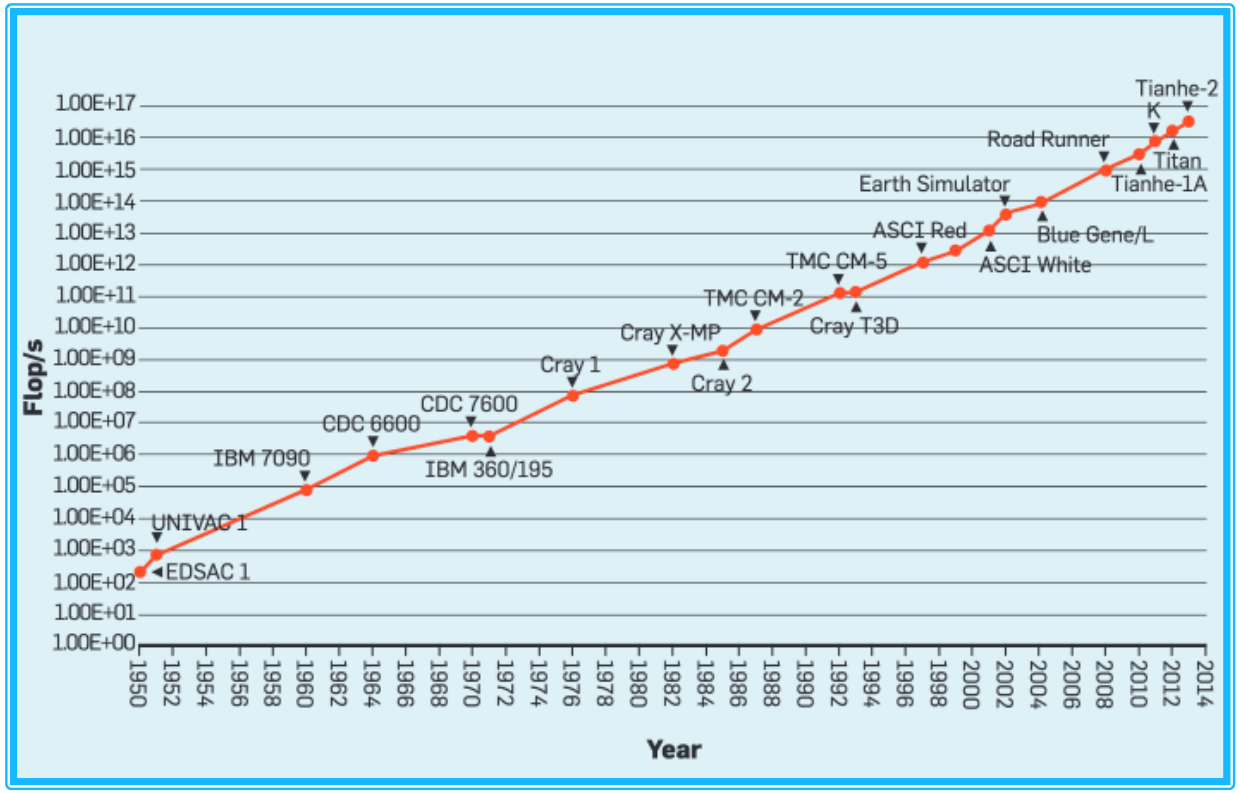
High Performance Computing and Energy

Part 1 of 5

Scientific advancement is requiring ever-increasing compute power

- The DOE Exascale Computing Project identifies the following challenge areas:
  - National Security Needs
  - Advances in Materials Science
  - New Energy Solutions
  - Advances in Healthcare
  - Predicting Severe Weather
  - Urban Science
  - High-Energy Physics
  - Astrophysics
  - Chemistry
  - Computer-Aided Design

HPC Computing Performance

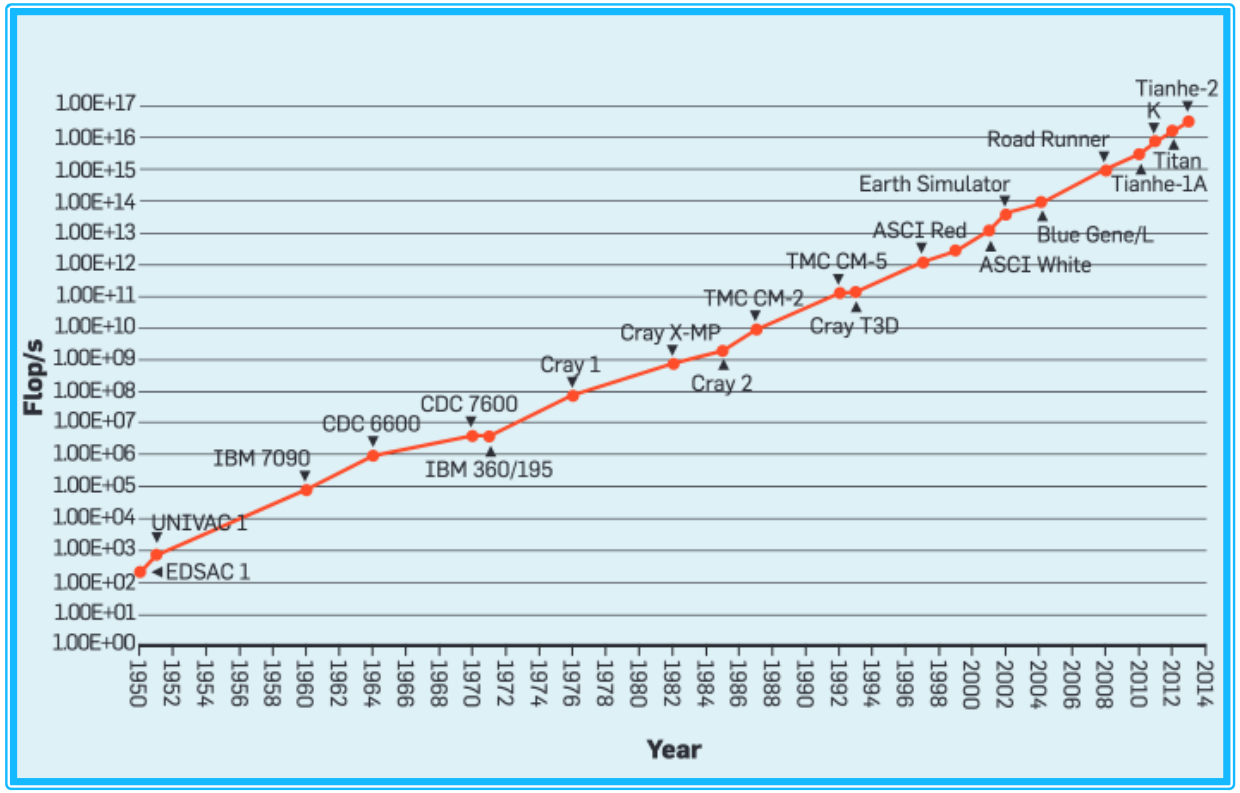


From Reed & Dongarra, 2015

Scientific advancement is requiring ever-increasing compute power

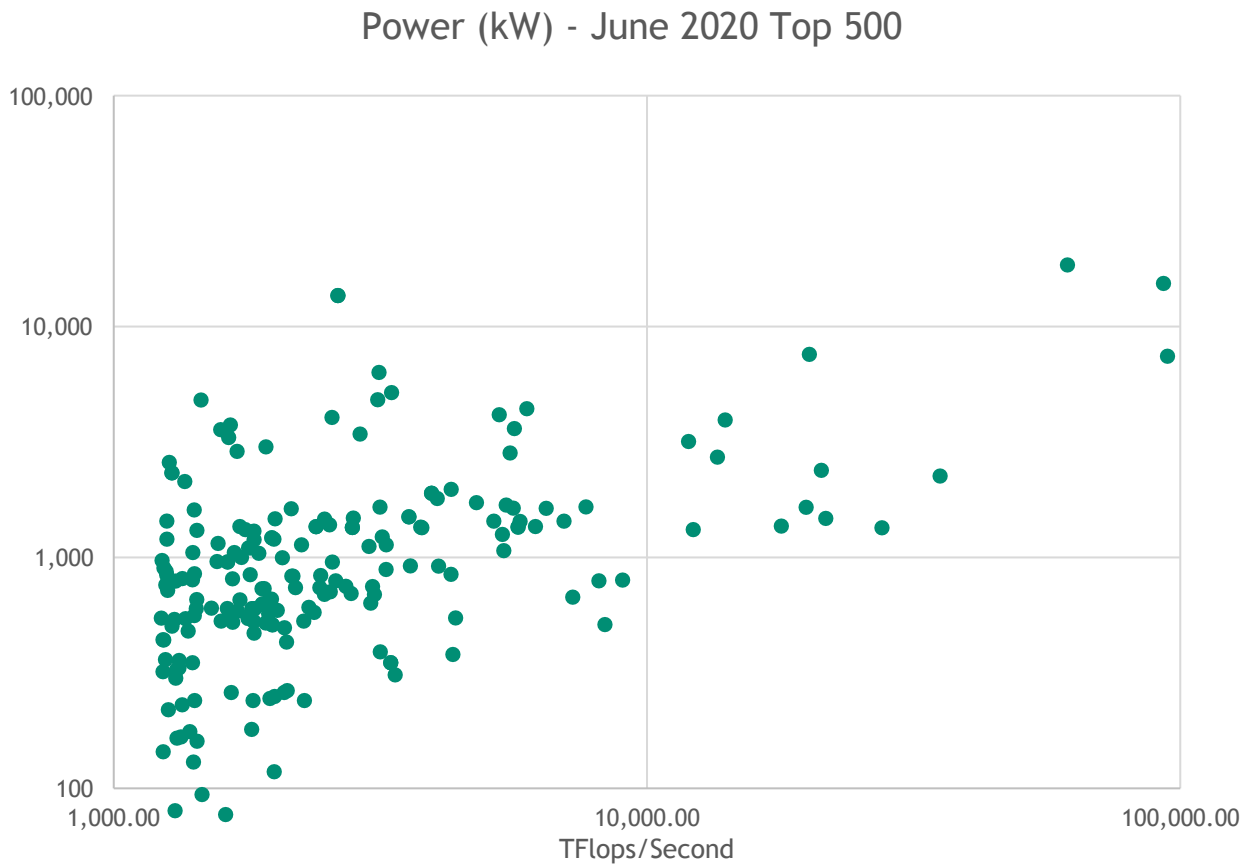
- Luckily, the enabling algorithms remain the same:
  - Dense Linear Algebra
  - Sparse Linear Algebra
  - Computation on Grids
  - Unstructured Grids
  - Finite Elements
  - Spectral Methods
  - Particle Methods
  - Monte Carlo
  - Graph Analytics

HPC Computing Performance



From Reed & Dongarra, 2015

What's stopping us?



1 exaflop = 1,000,000 teraflops

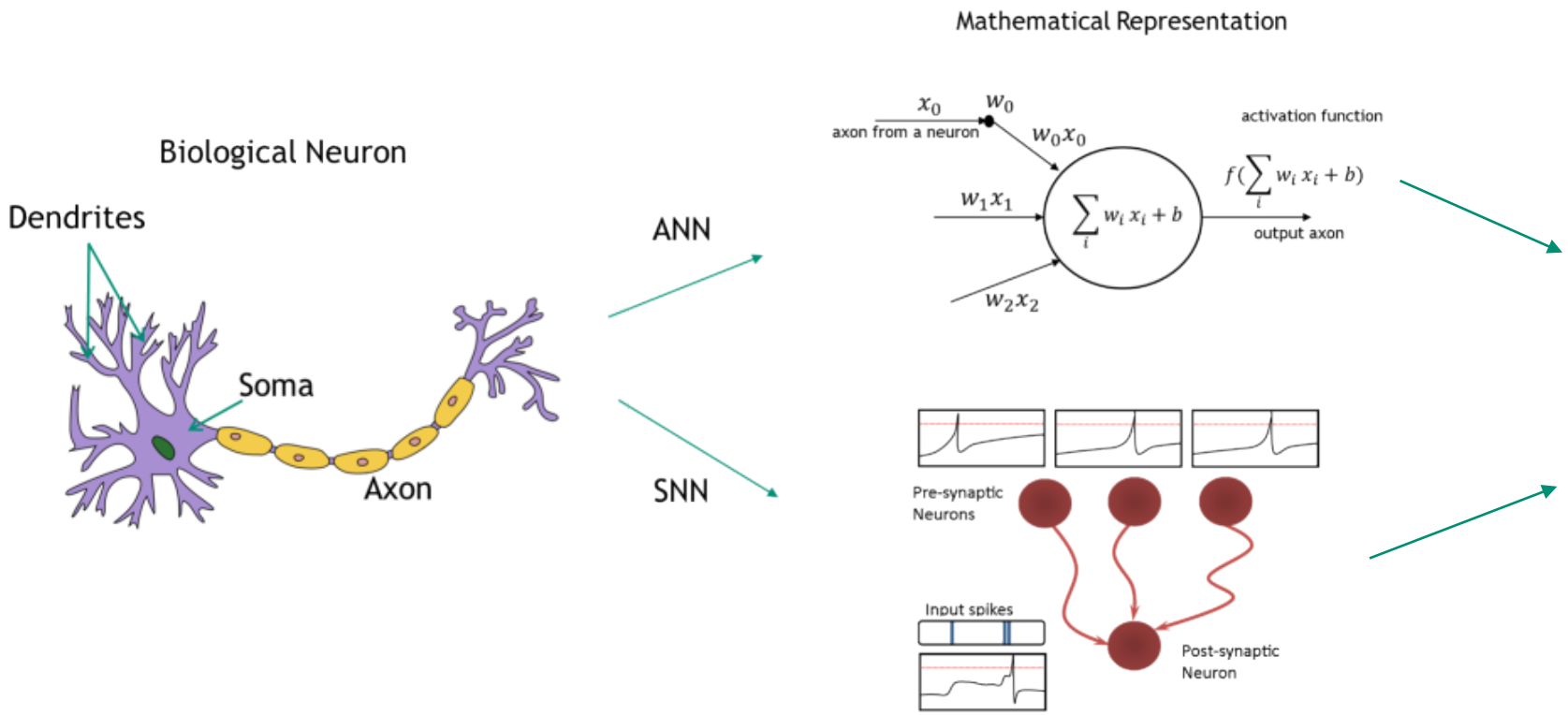
- Supercomputers are increasingly limited by power consumption.
- Exascale systems are forecasted to begin in the 100,000's of kW.
  - That's enough electricity to power around 80,000 US homes for a year.
  - Rome, GA has a population of 36,332.
- Are there low-power alternatives that can solve the same problems for which we currently require traditional high-performance computing?

Neural-Inspired Computing

Part 2 of 5

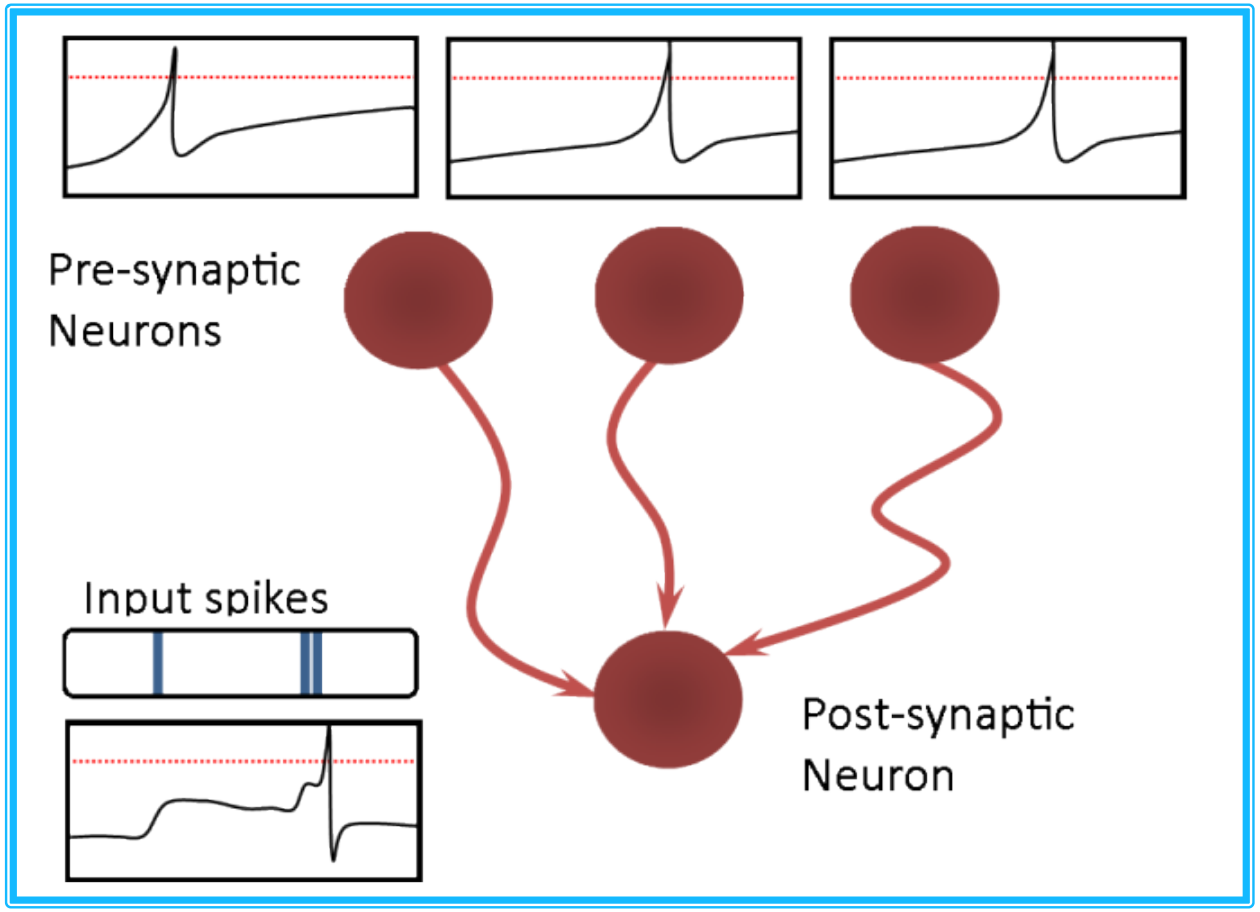
Neural-Inspired Computing

- What is neural-inspired, neuromorphic, or brain-inspired computing?
  - Fundamental notion of taking inspiration from how the brain performs computation.



Spiking Neural Networks

- Subclass of Artificial Neural Network
- Neurons compute their own state independently, possibly asynchronously
- Each neuron integrates incoming information into a ‘potential’
- If ‘potential’ reaches a predetermined threshold, the neuron alerts connected neurons
- Neuron communication is single-state signals (spikes)
- A time delay for spike propagation can be included
- Enables event-driven computation



Spiking Neural Networks – Neuron Dynamics

Generically, a discrete-time leaky-integrate-and-fire neuron is well-modeled by simulators and neuromorphic hardware.

For random draw η and weights $w_{i,j}$, delays $d_{i,j}$, initial voltages $V(0)$, probability of fire P_i, and initial action potentials $x(0)$ being algorithm dependent:

$$\widehat{V}_i(t+1) = V_i(t) + \sum_j w_{i,j} x_j\left(t - d_{i,j} + 1\right)$$

$$x_i(t+1) = \begin{cases} 1, & \hat{V}(t+1) > V_i^* \text{ and } \eta_{i,j} < P_i \\ 0, & \text{otherwise} \end{cases}$$

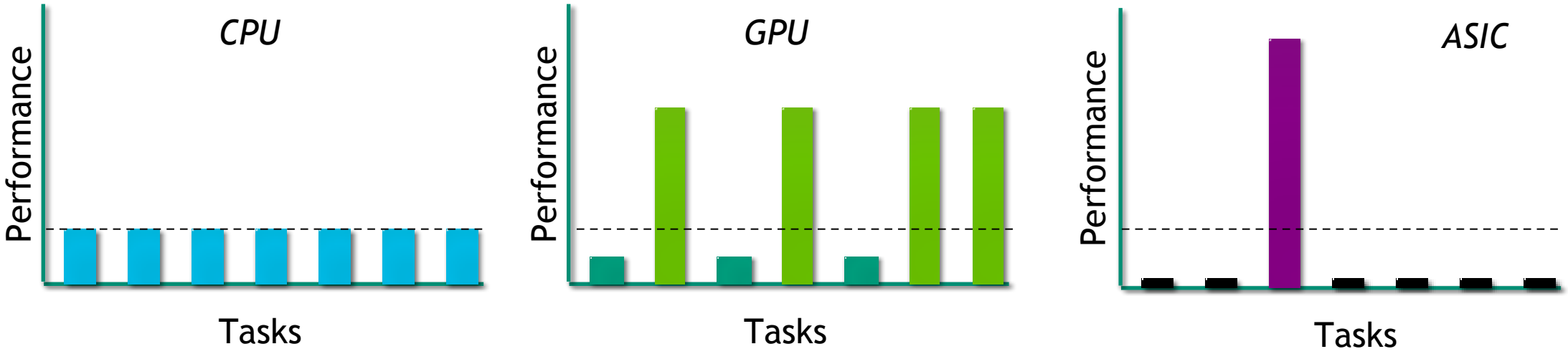
$$V_i(t+1) = \begin{cases} \tau_i \hat{V}_i(t), & x_i(t+1) = 0 \\ 0, & x_i(t+1) = 1 \end{cases}$$

Each neuron processes these functions at **every time step** in **perfect parallel**.

If you can take your algorithm and formulate it
as a network of these independent processes, it can run on neuromorphic.

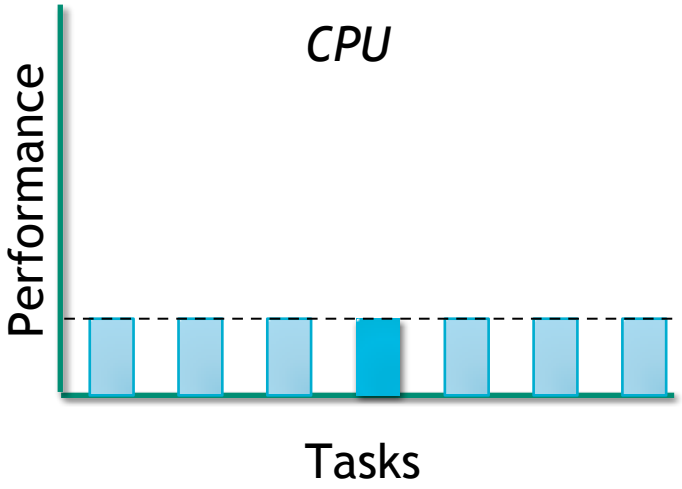
Neuromorphic and the “No Free Lunch” Theorem

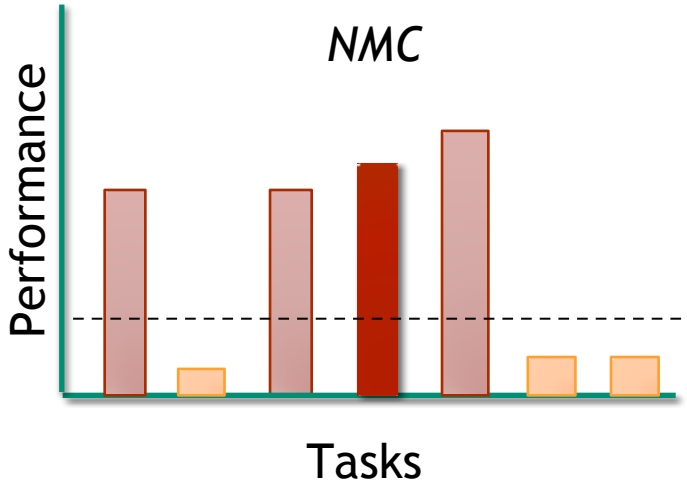
No Free Lunch (Wolpert & Macready, 1997) Theorem implies that optimizations to improve some calculations will make other calculations more expensive

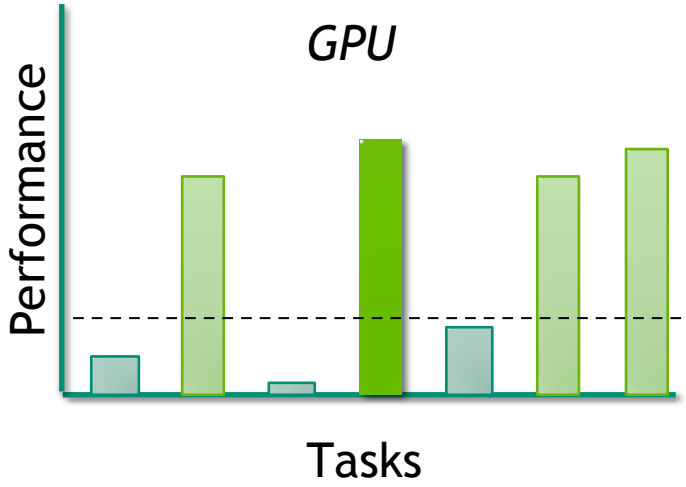


Customizing architecture to be good at one thing makes it worse at everything else

The problem with deep learning …

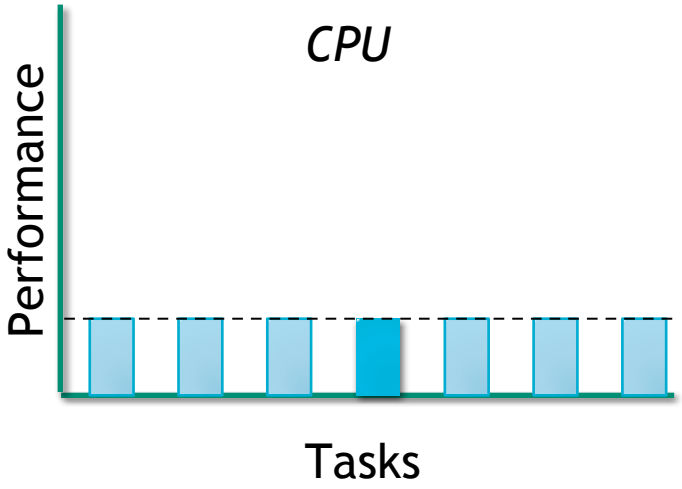


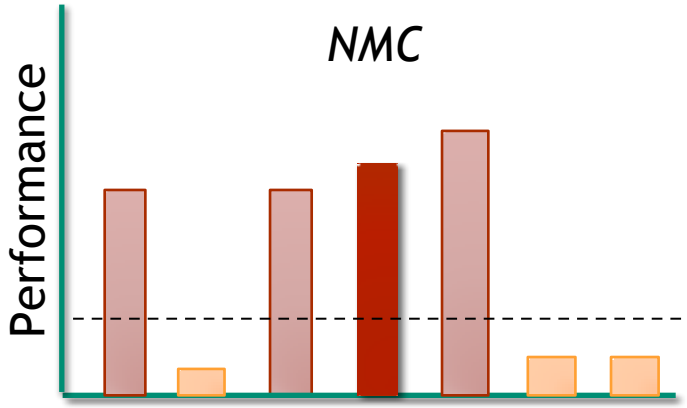


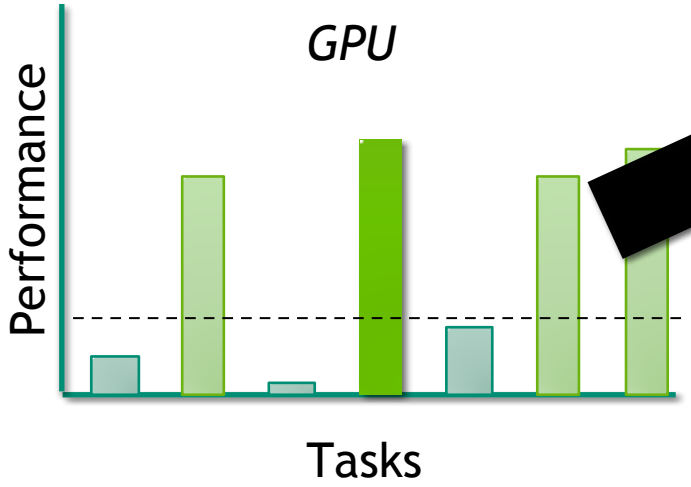


- Yay! Deep learning and neural networks look like a pretty natural fit for neuromorphic computing (NMC)!
- BUT… deep learning and neural networks are also a *very* natural fit to GPUs. Perhaps even a better fit…
- We shouldn't be surprised… GPUs are more mature technology and deep learning revolution is largely due to GPUs being good at them

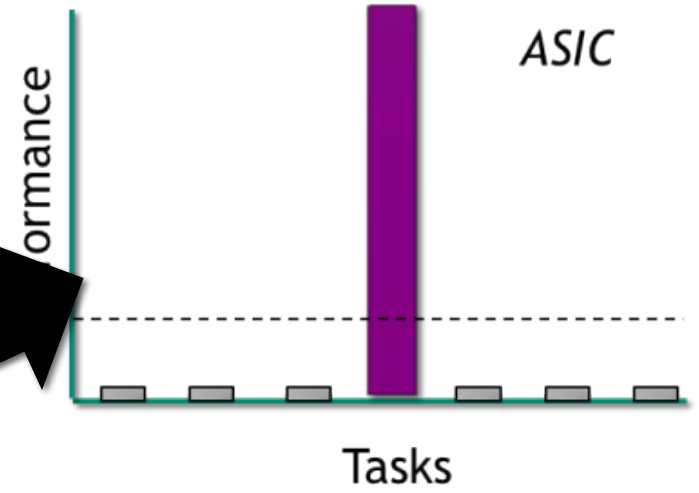
The problem with deep learning …



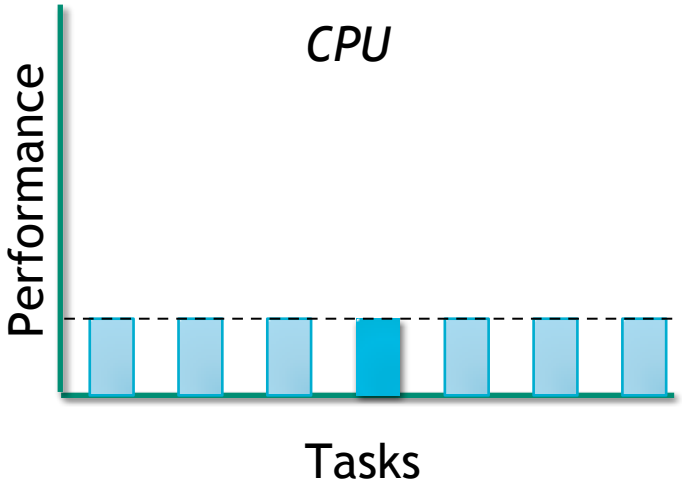


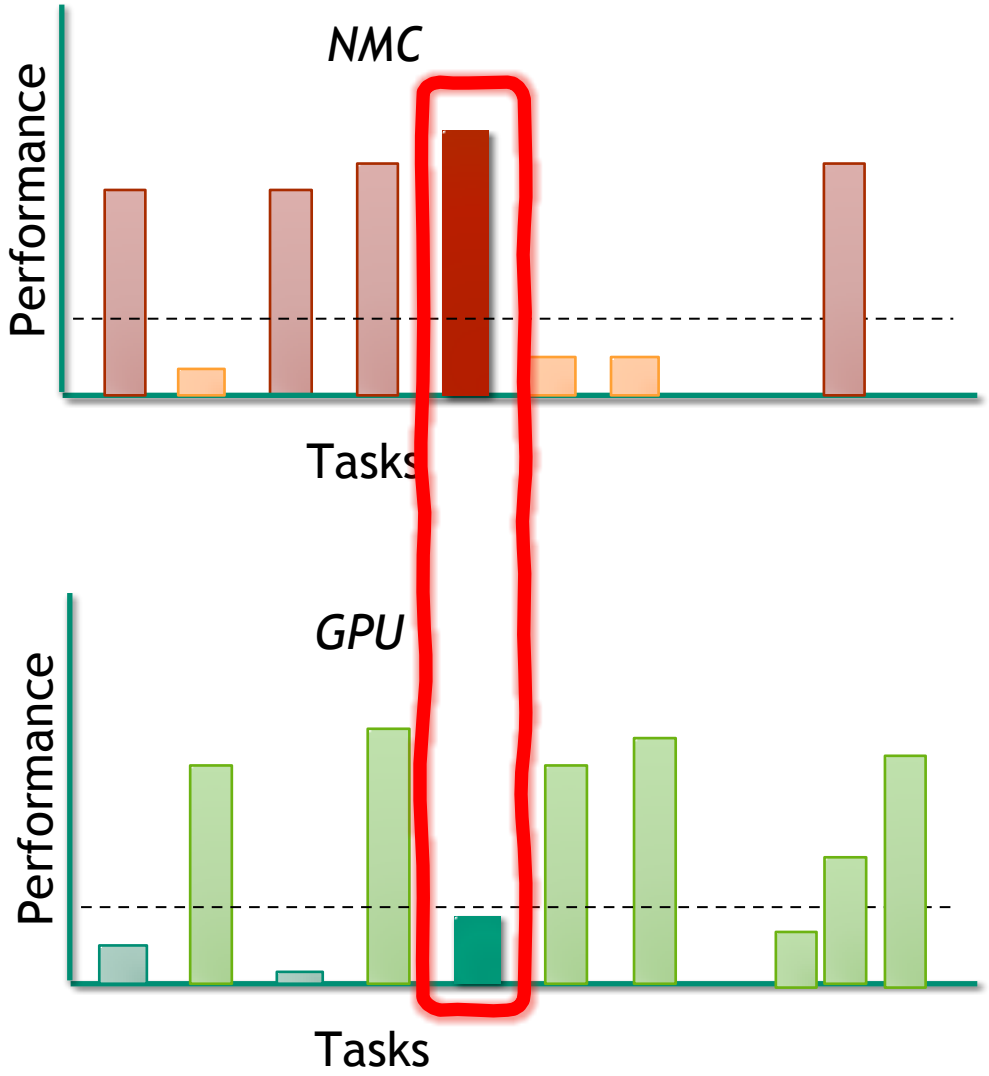


Even bigger problem… neural network accelerators



Where do GPUs not offer an advantage?

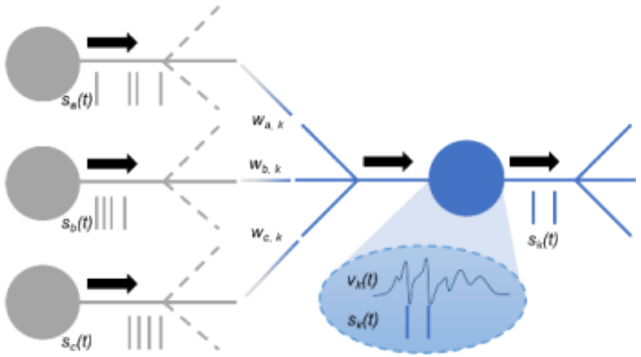




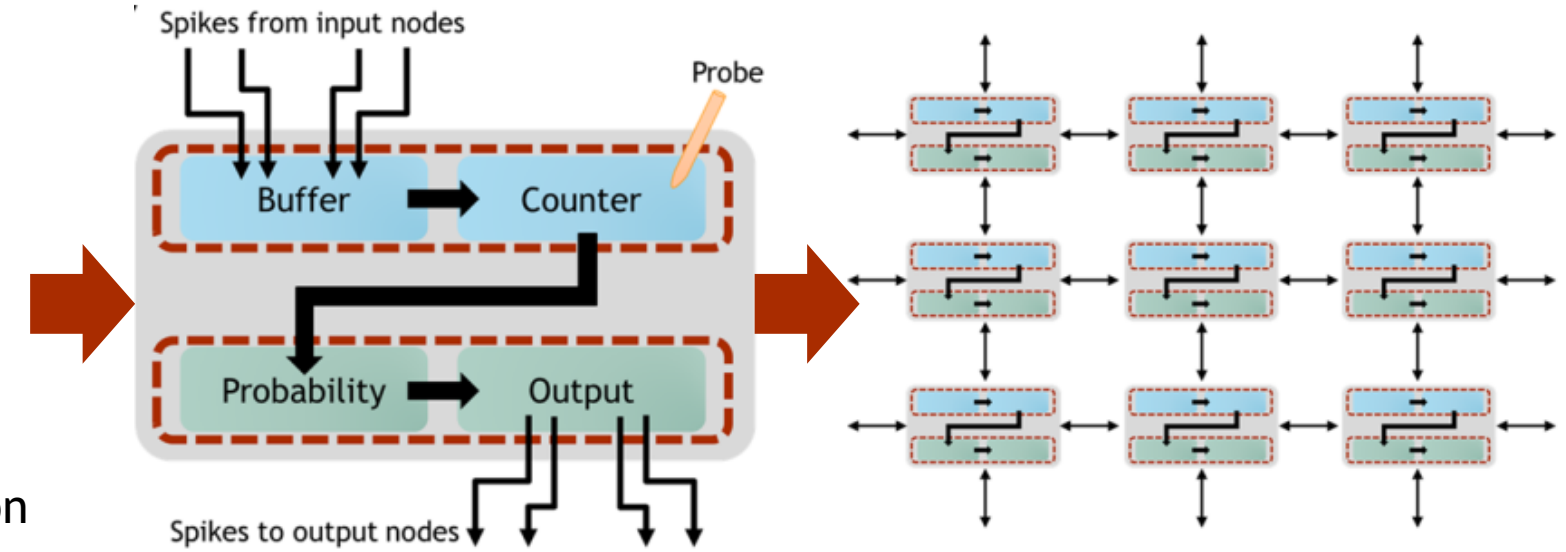
Neural Random Walks and Applications

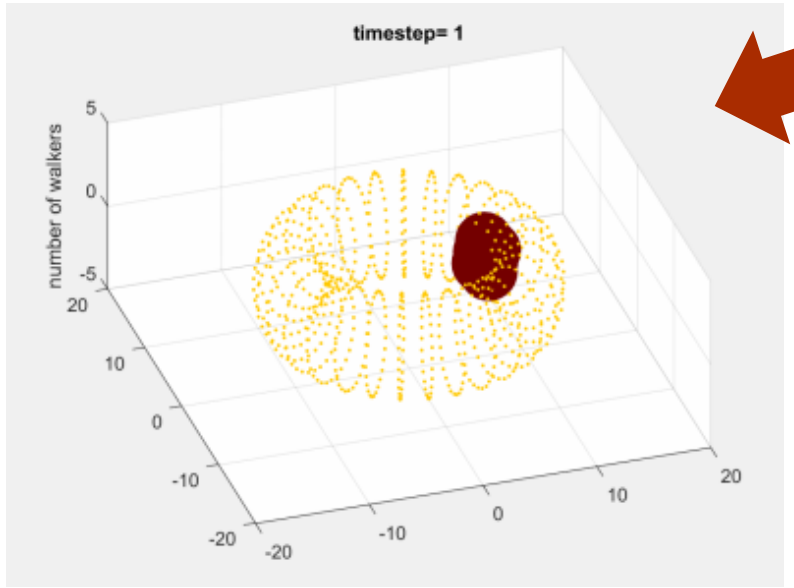
Part 3 of 5

Neuromorphic algorithm can simulate random walks



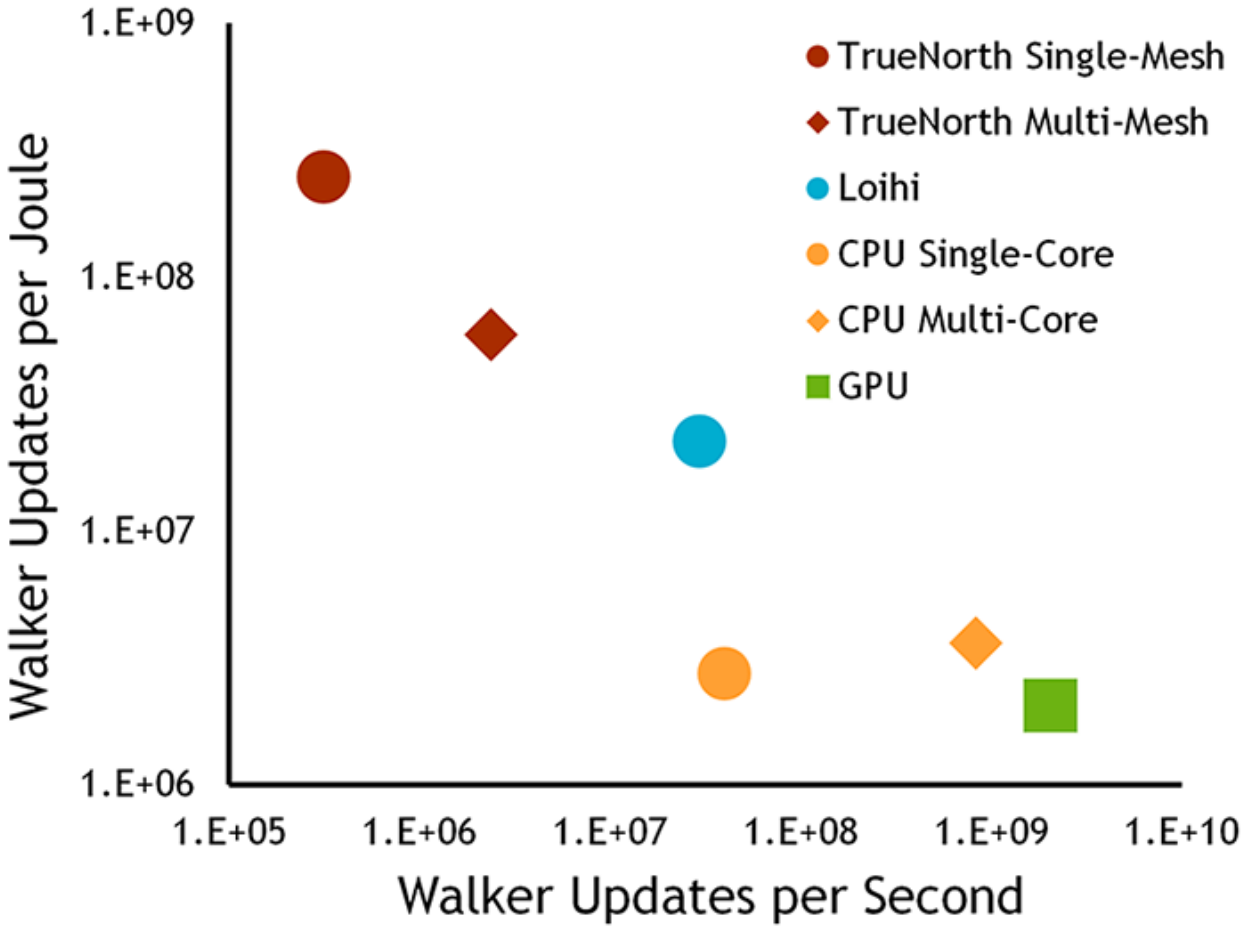
Leaky Integrate and Fire Neuron





Smith et al., in review 2021

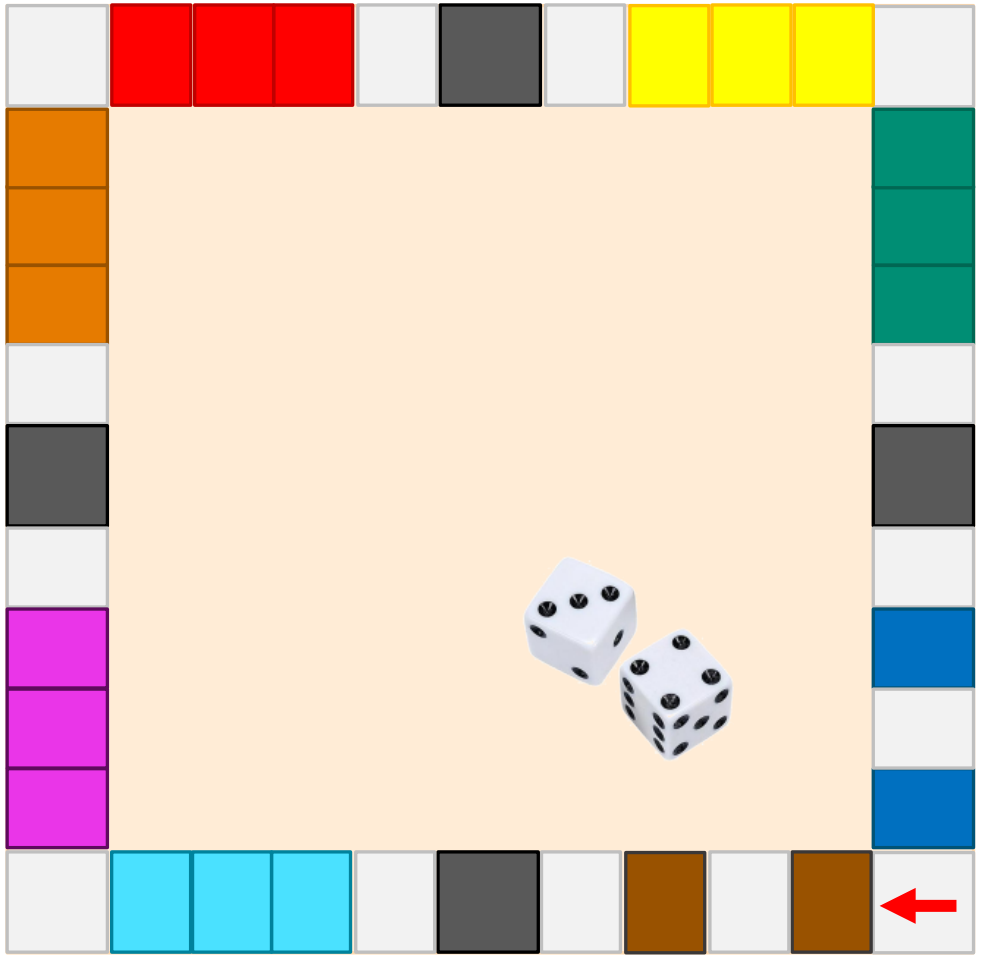
Neuromorphic Efficiency



Smith et al., in review 2021

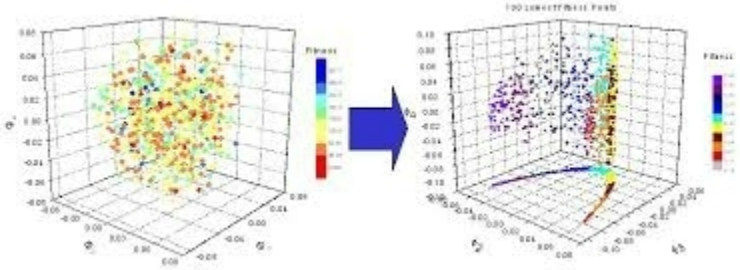
Random Walks and Monte Carlo

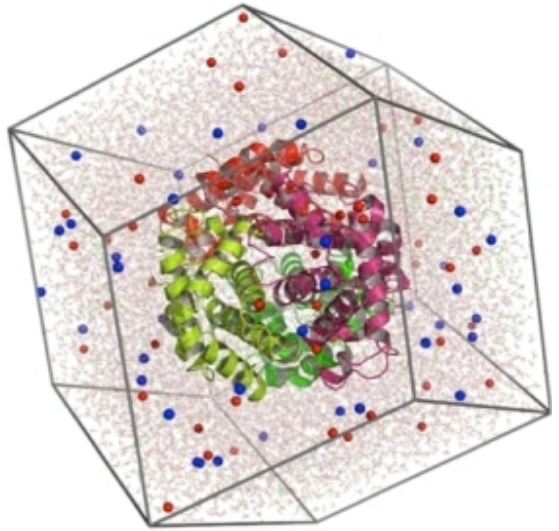
- We want to determine the probability of landing on any given space in 1 trip around the board.
- To make a best estimate, we start a token and record the spaces it lands on in one trip around the board.
- We estimate the probability of each space by repeating the process several times, recording the spaces for many different instances of a token around the board, and averaging.
- The process of gathering the path of many tokens around the board is called *sampling*, and the use of averaging makes our answer a *Monte Carlo* estimate.



Monte Carlo Approximations of PDE Solutions

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





Math: What PDEs can neuromorphic random walks be useful for?

Class of Partial Integro-Differential Equations:

$$\frac{\partial}{\partial t}u(x,t) = \frac{1}{2}\sum_{i,j}(\sigma\sigma^{T})_{i,j}(x,t)\frac{\partial^{2}}{\partial x_i \partial x_j}u(x,t) + \sum_{i}\mu_i(x,t)\frac{\partial}{\partial x_i}u(x,t)$$
$$+ r(x,t)\int_{\mathbb{R}^d} u(x+\eta(x,t,z),t)\,\nu(x,t,z)\,dz - r(x,t)\,u(x,t)$$
$$+ q(x,t)u(x,t) + f(x,t), \qquad x \in \mathbb{R}^d,\ t \in [0,\infty).$$

Stochastic Process:

NMC Hardware Simulates This Stochastic Process

$$dX(t) = \mu(X(t),t)\,dt + \sigma(X(t),t)\,dB(t) + \int_{\mathbb{R}^d}\eta(X(t),z)\,dN(t,z,X(t)).$$

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

Monte Carlo Approximates This Expectation

$$u(x,t) = \mathbb{E}\left[g(X(t))\exp\left(\int_0^t r(X(s),s)\,ds\right) + \int_0^t f(X(s),s)\exp\left(\int_0^s r(X(\ell),\ell)\,d\ell\right)ds \;\middle|\; X(0)=x\right].$$

Neural MC algorithm can run wide range of stochastic processes

$$\frac{\partial}{\partial t}\phi(x,t) = \frac{1}{2}\sum_{i,j}(\sigma\sigma^{\top})_{ij}(x,t)\frac{\partial^2}{\partial x_i \partial x_j}\phi(x,t) + \sum_i \mu_i(x,t)\frac{\partial}{\partial x_i}\phi(x,t)$$

$$+ \int_{\mathbb{R}^d}\lambda(x,t)\,p(y;x,t)\,\phi(y,t)\,dy - \lambda(x,t)\,\phi(x,t)$$

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

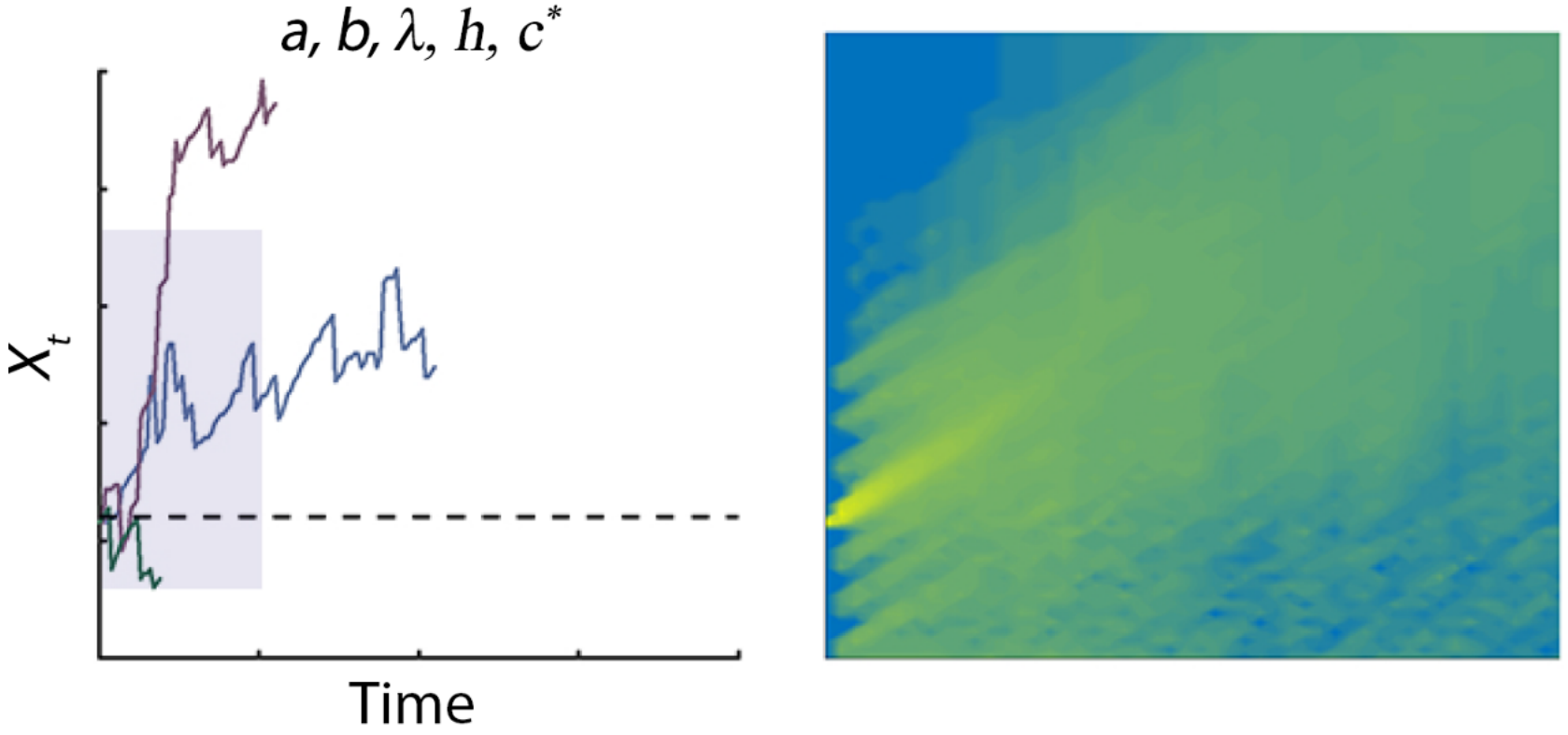
$$dX(t) = \mu(X(t),t)\,dt + \sigma(X(t),t)\,dB(t) + \int_{\mathbb{R}^d} z\,N(dz,dt;X(t),t).$$

Drift

Diffusion

Jumps

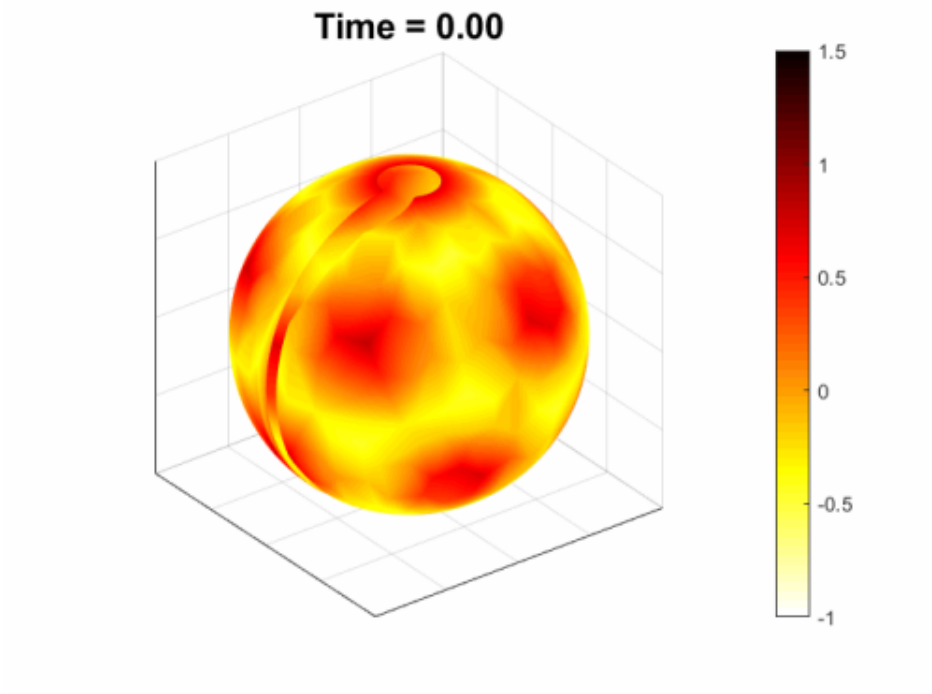
Death

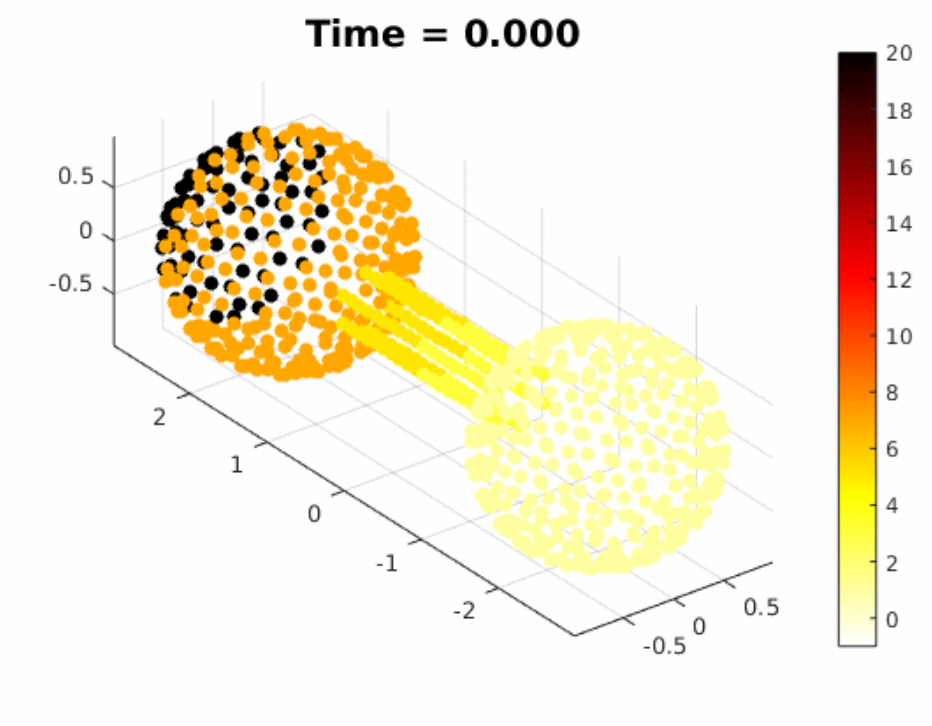


Smith et al., in review 2021

Some Physics Examples

- Heat transport on surface of sphere and barbell
  - Not a trivial diffusion surface, complex stochastic process and transition probabilities
  - Implemented sphere on Loihi; barbell in simulation

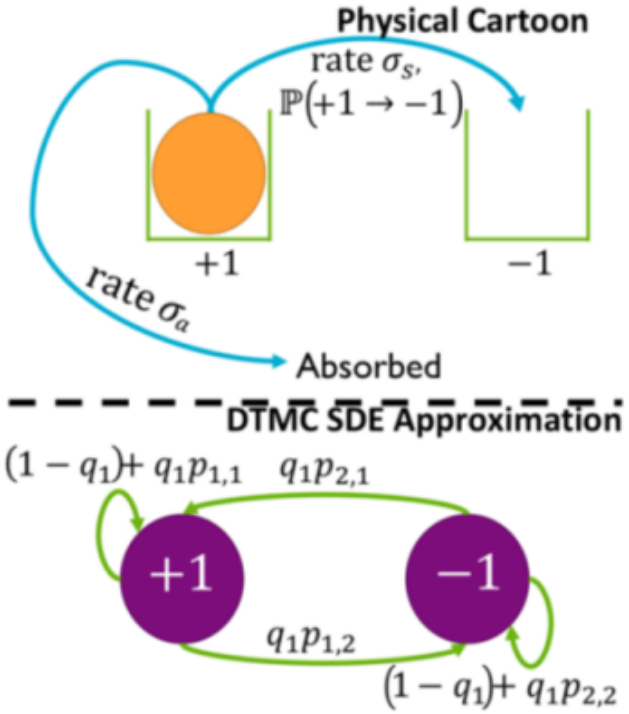


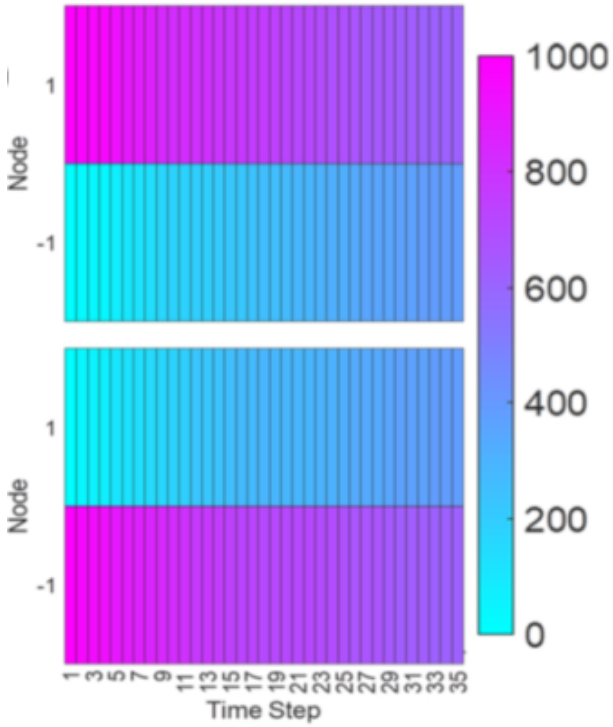


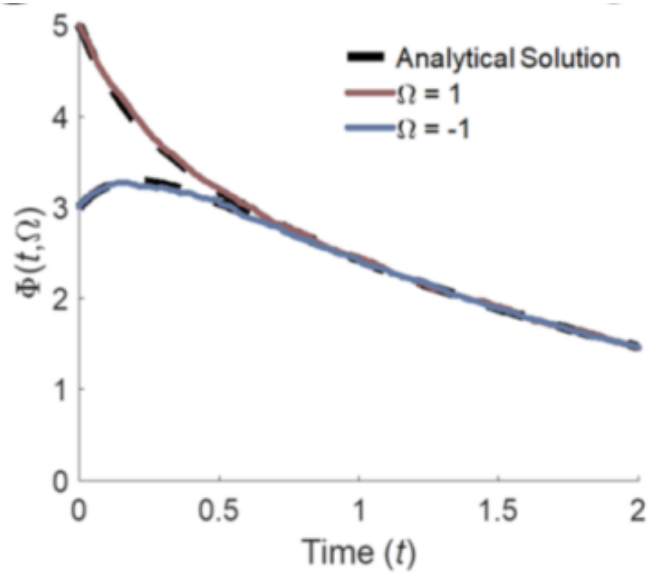
Smith et al., in review 2021

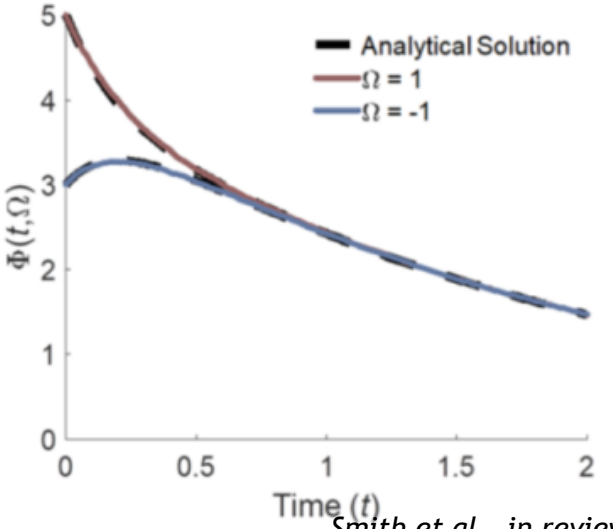
Some Physics Examples

- Boltzmann state transition
  - Particle can exist in 2 states (+1 or -1) or be absorbed.
  - Implement as simple stochastic process on Neuromorphic.





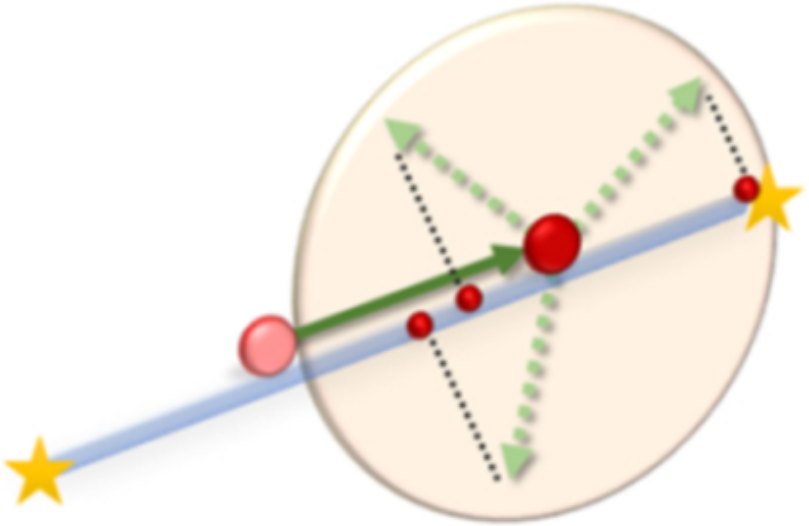


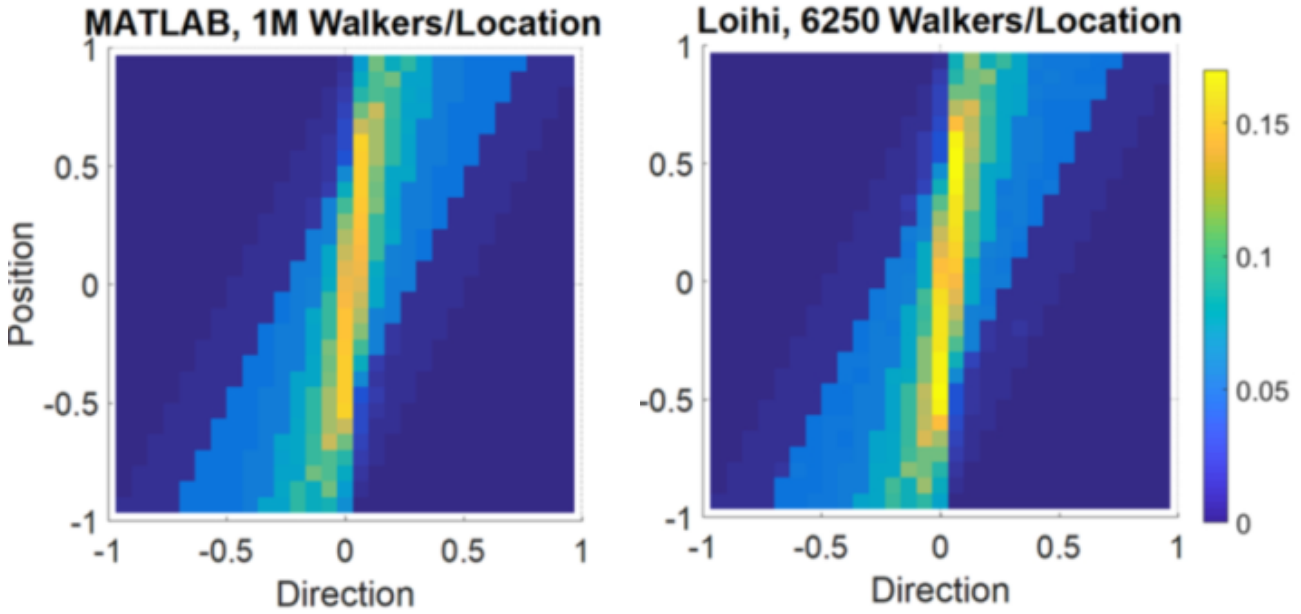


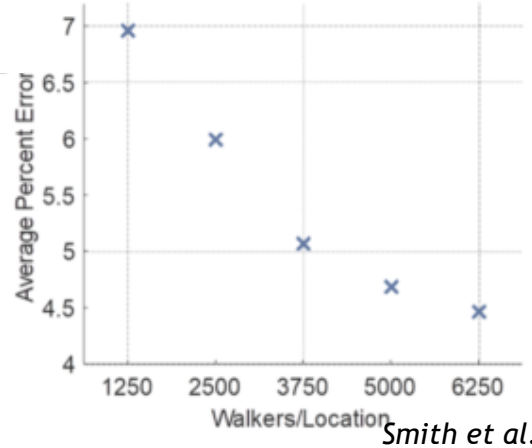
Smith et al., in review 2021

Some Physics Examples

- 1D particle transport
  - We allow directions in $[-1,1]$.
  - Particle moves with a fixed velocity.
  - Track total flux of particles as a function of both position and direction.





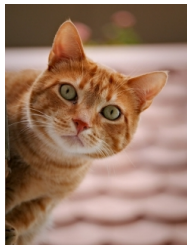


Smith et al., in review 2021

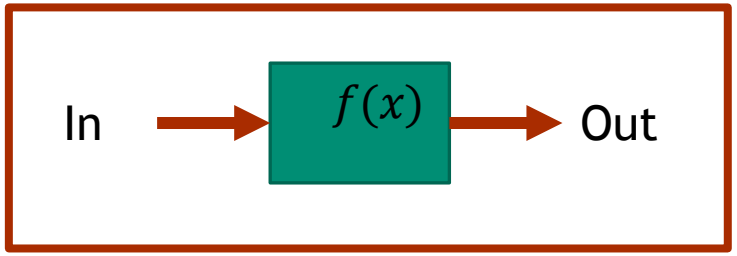
Learning Random Walks – An Inverse Problem

Part 4 of 5

Machine Learning – The General Idea

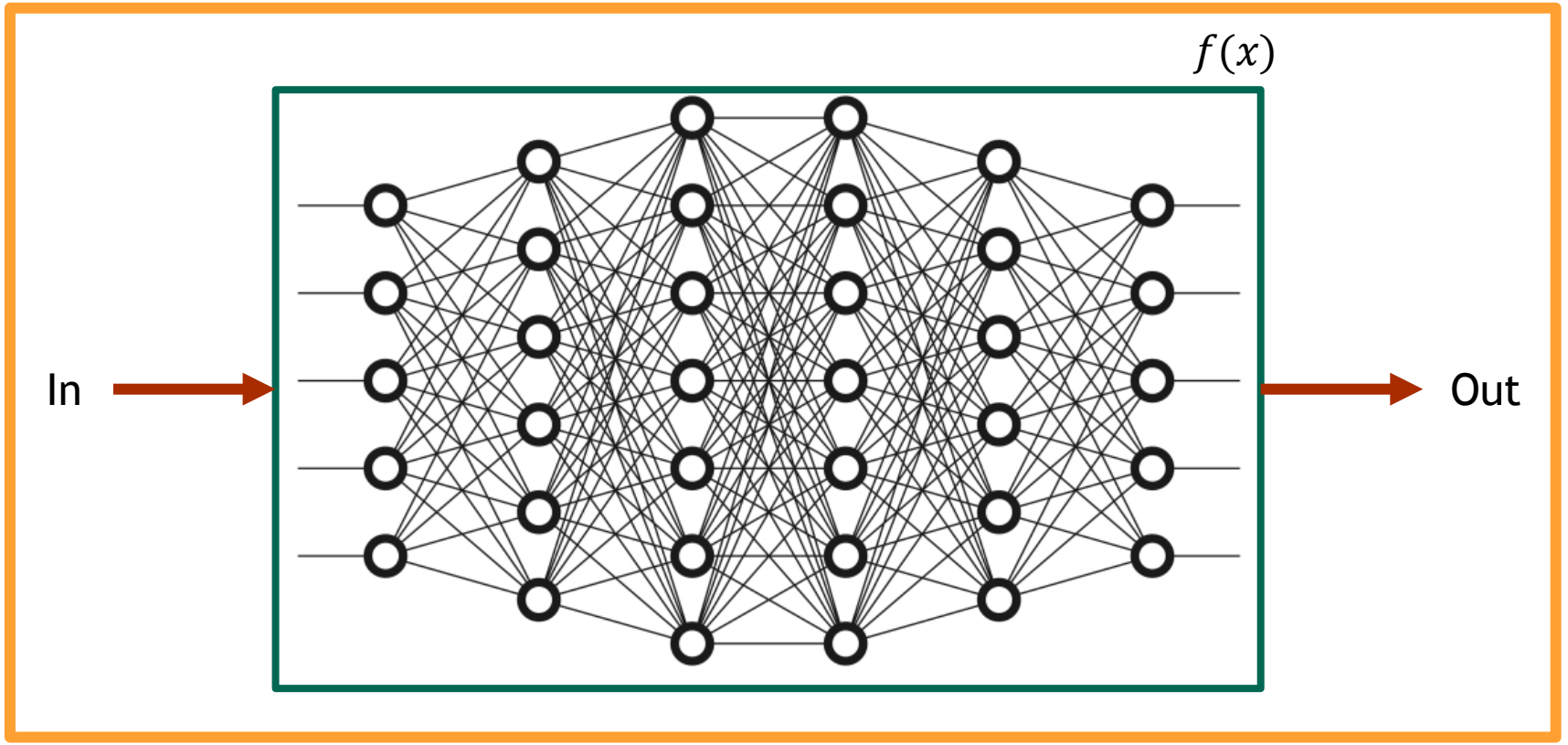


Real Life



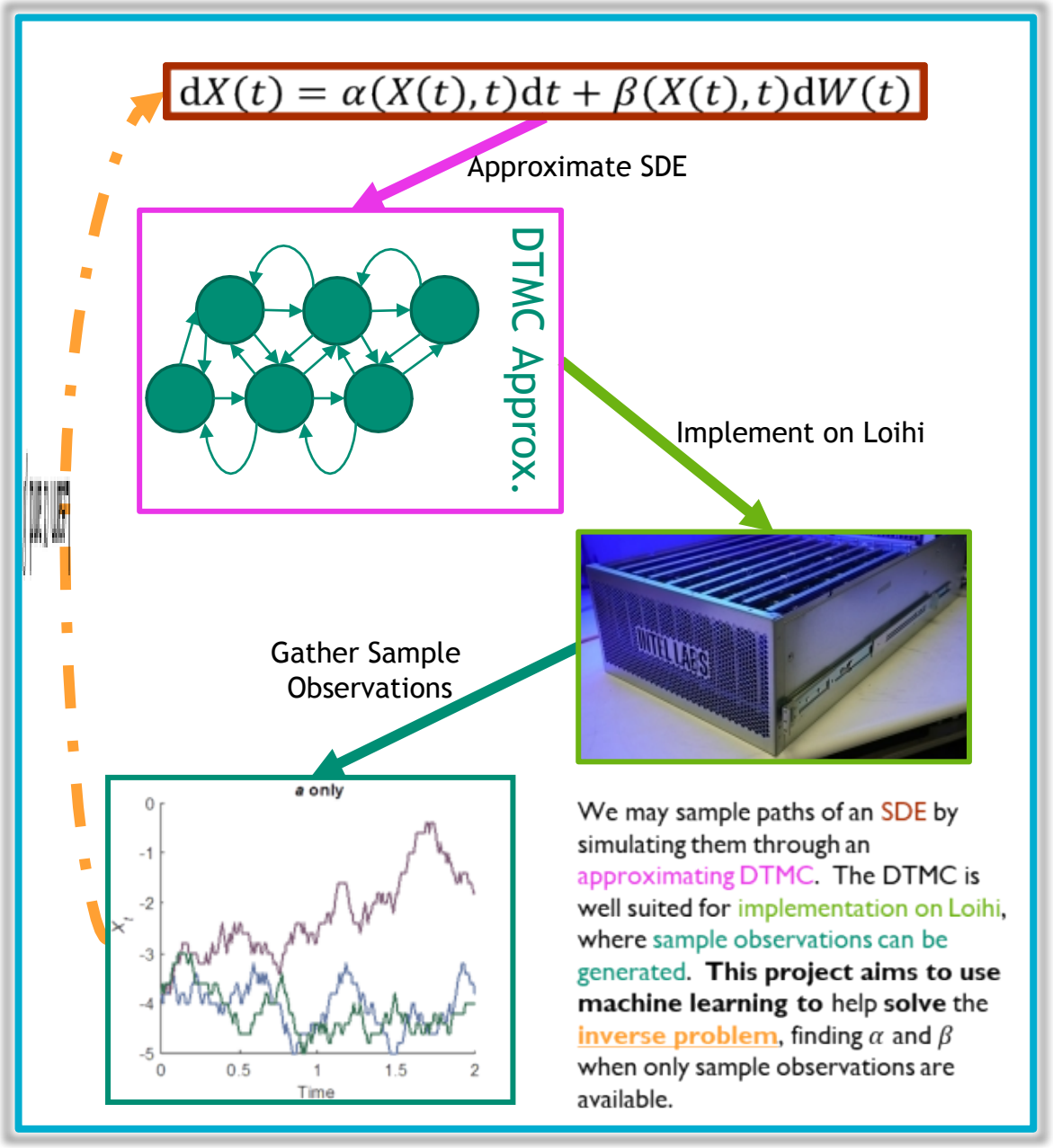
cat

Deep Learning



Learning Random Walks – An Overview

- This current project has two goals:
  - Can Machine Learning (ML) demonstrate success on an inverse problem?
  - Are simulated data from neuromorphic machines 'good enough' to use for scientific computing purposes?

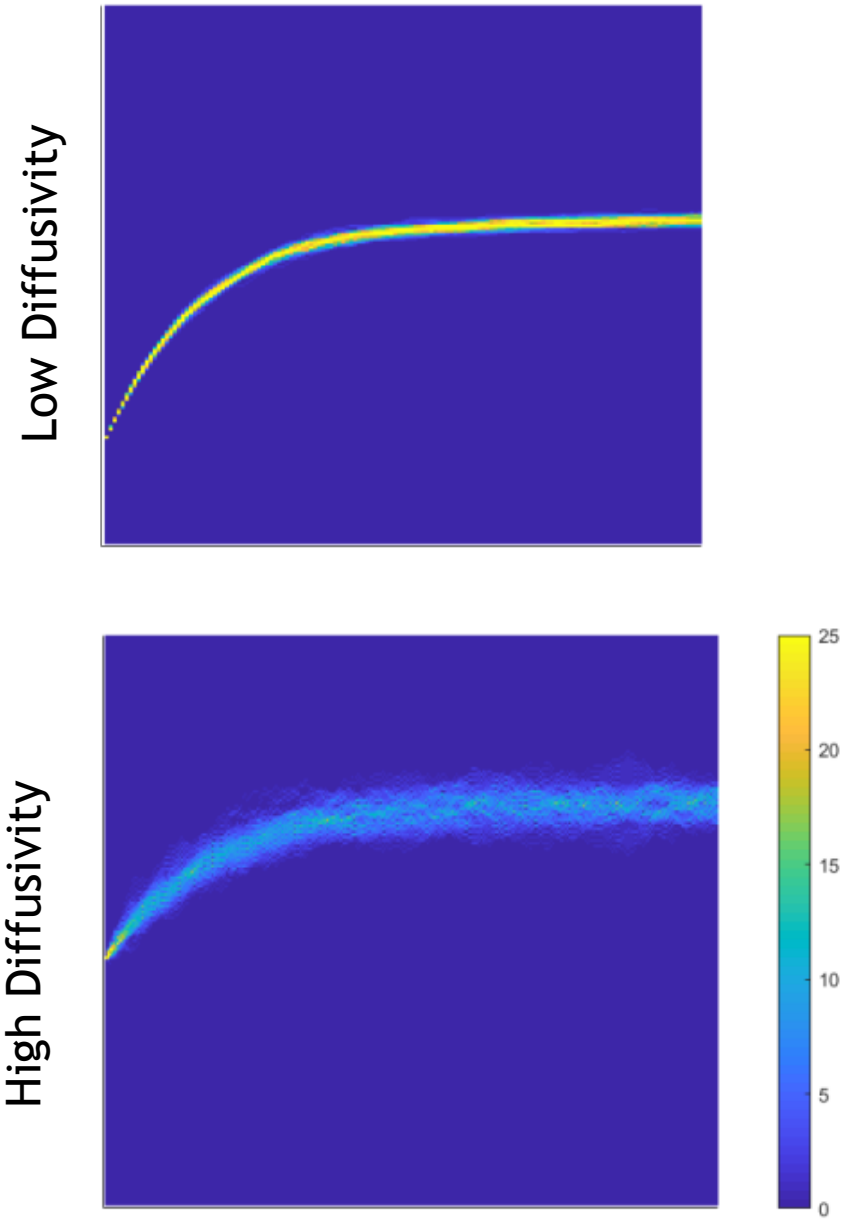


The Ornstein-Uhlenbeck Equation

- To limit the focus, we consider the known equation:

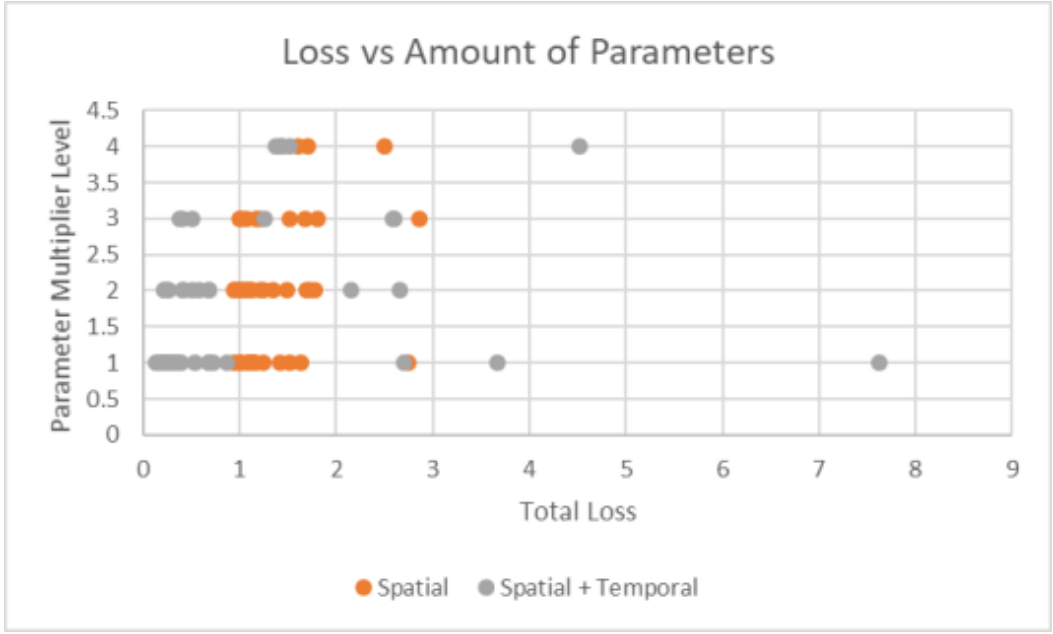
$$\mathrm{d}X(t) = -k(X(t) - z)\mathrm{d}t + \sqrt{2D}\mathrm{d}W(t).$$

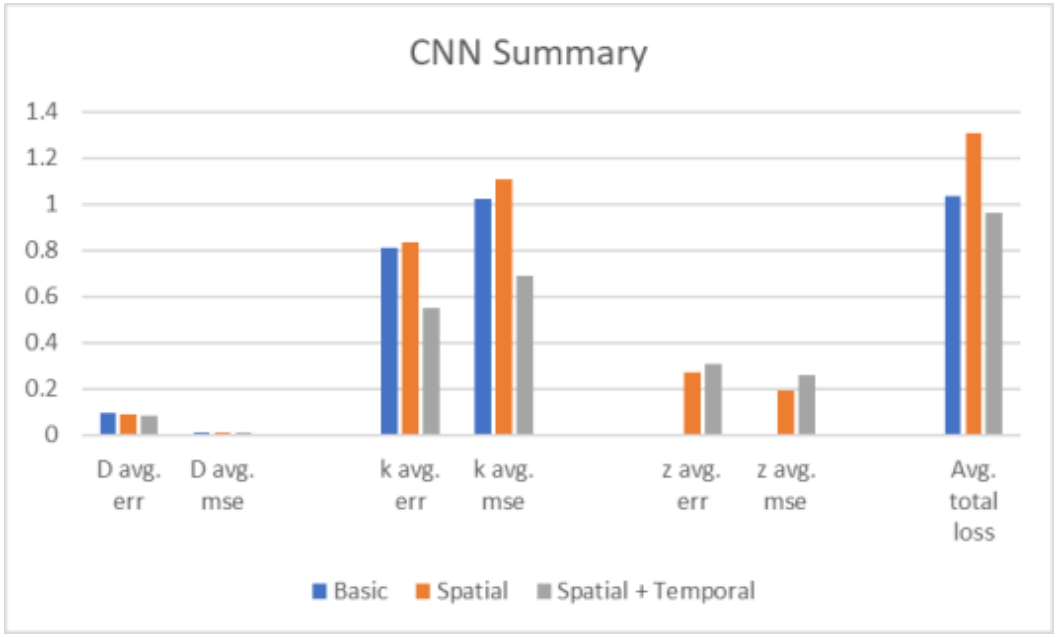
  - Used to model
    - **Brownian particle in a harmonic well;**
    - **Hookean spring;**
    - **Biology applications - in cell transport;**
    - **Optical tweezers.**
  - k - spring constant
  - z - center position
  - D - diffusivity
- This is an easy problem to start with since analysis methods are known!



Results of Learning

- We used a particular kind of deep learning – a convolutional neural net (CNN) – to learn the parameters given generated neuromorphic random walk data.
- Three configurations have been tested:
  - Basic – no physical information provided to the CNN.
  - Spatial – information about the spatial window is given.
  - Spatial + Temporal – all the spatial information plus information about the time scale.





Non-Academic Career Paths in Mathematics and Computer Science

Part 5 of 5

Non-Academic Career Paths in Mathematics and Computer Science

- Government
  - National labs
  - Three letter orgs like the NSA
- Industry
  - Google, Facebook, Intel, IBM
  - Corporate owned R&D, like PARC
  - Institutional research roles
- Finance
  - Actuarial
  - Data management
  - Modeling/Risk Modeling
- Many, many more!

Neural Computing at Sandia

NERL

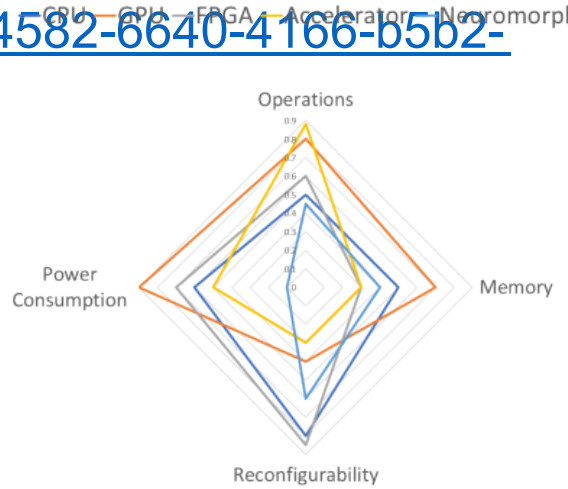
Neural Exploration & Research Lab (NERL)

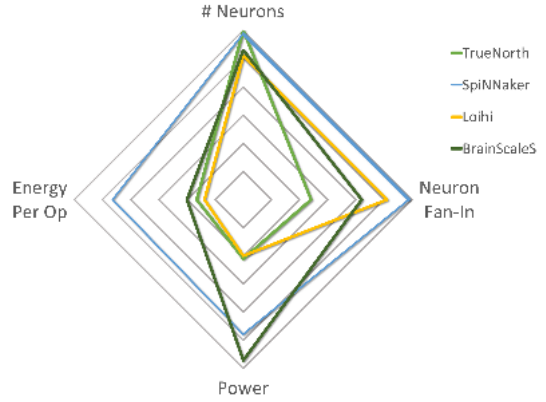
Enables researchers to explore the boundaries of neural computation

- The research conducted in the lab evaluates what is possible with neural hardware and software for national security benefit and the advancement of basic research

Consists of a variety of neuromorphic hardware & neural algorithms providing a testbed facility for comparative benchmarking and new architecture exploration

Official Lab Partnering Service capability advocating Sandia as a neuromorphic center of excellence: https://www.labpartnering.org/facility/3dd34582-6640-4166-b5b2-



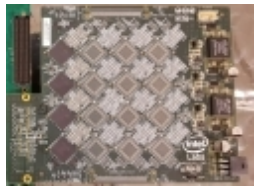


Neuromorphic Hardware

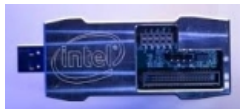
World class research capability offering unique ability to research aspects of many different neuromorphic architectures

Intel Loihi 50M

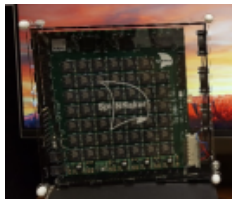
Intel Loihi Nahuku Bay (8)



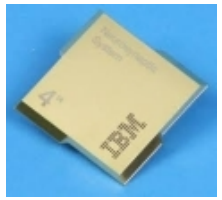
Intel Loihi Kapoho Bay USB



SpiNNaker 48 Node Board



IBM TrueNorth*



IBM TrueNorth NS16e*

GraphCore

Groq

Intel Neural Compute Stick

Google Coral

Google EdgeTPU

GyrFalcon

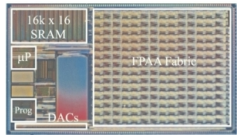
EtaCompute Tensai

Cognimem CM1K

KnuPath Hermosa

Inilabs DAVIS 240C DVS

Georgia Tech FPAA



SNL STPU on FPGA

Xilinx PYNQ FPGA

Nengo FPGA

Nvidia Jetson TX1

Nvidia Jetson Nano

GPU Workstations

*Remote access

Thanks!

- Sandia Neural PDE team:

  Brad Aimone, Aaron Hill, Leah Reeder, Ojas Parekh, William Severa, Brian Franke, Rich Lehoucq

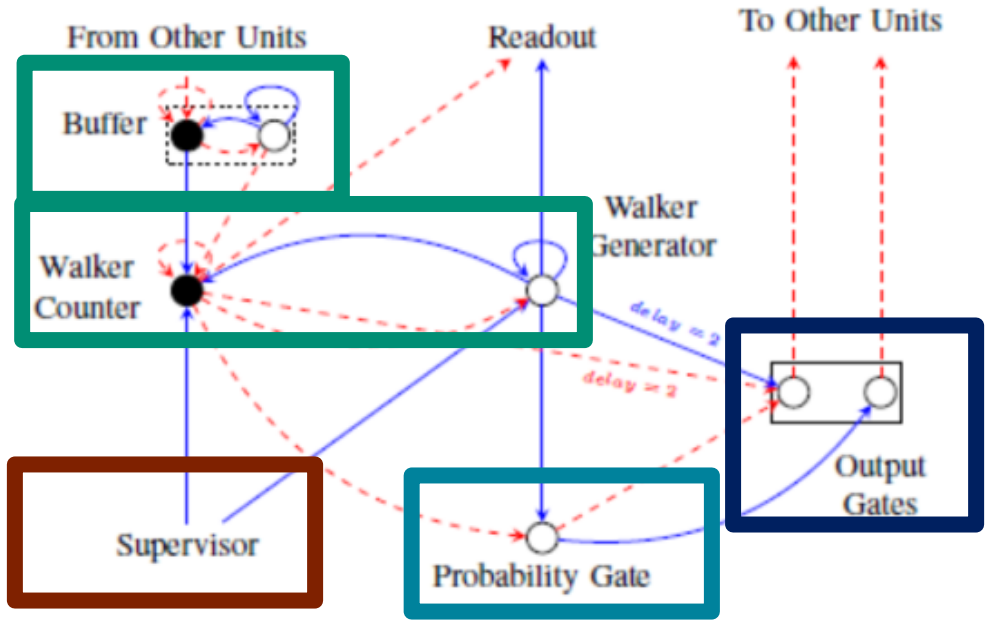
- Sandia Learning Random Walks team:
  - Brad Aimone, William Severa, Rich Lehoucq

- Neuromorphic Hardware:
  - Loihi Access: Craig Vineyard (SNL), Suma Cardwell (SNL), Intel INRC
  - TrueNorth Access: Lawrence Livermore National Laboratory

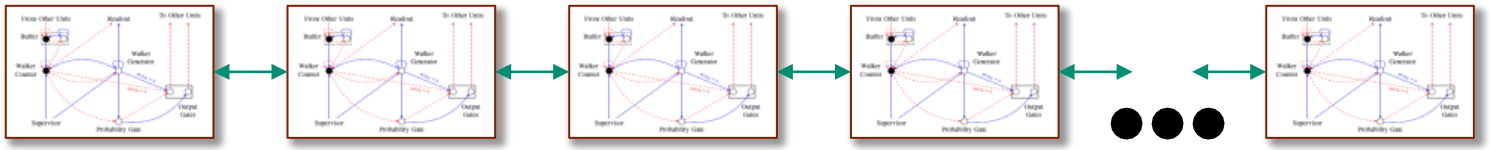
- Funding
  - Sandia Laboratory Directed Research and Development
  - DOE Advanced Simulation and Computing

Additional Slides and Information

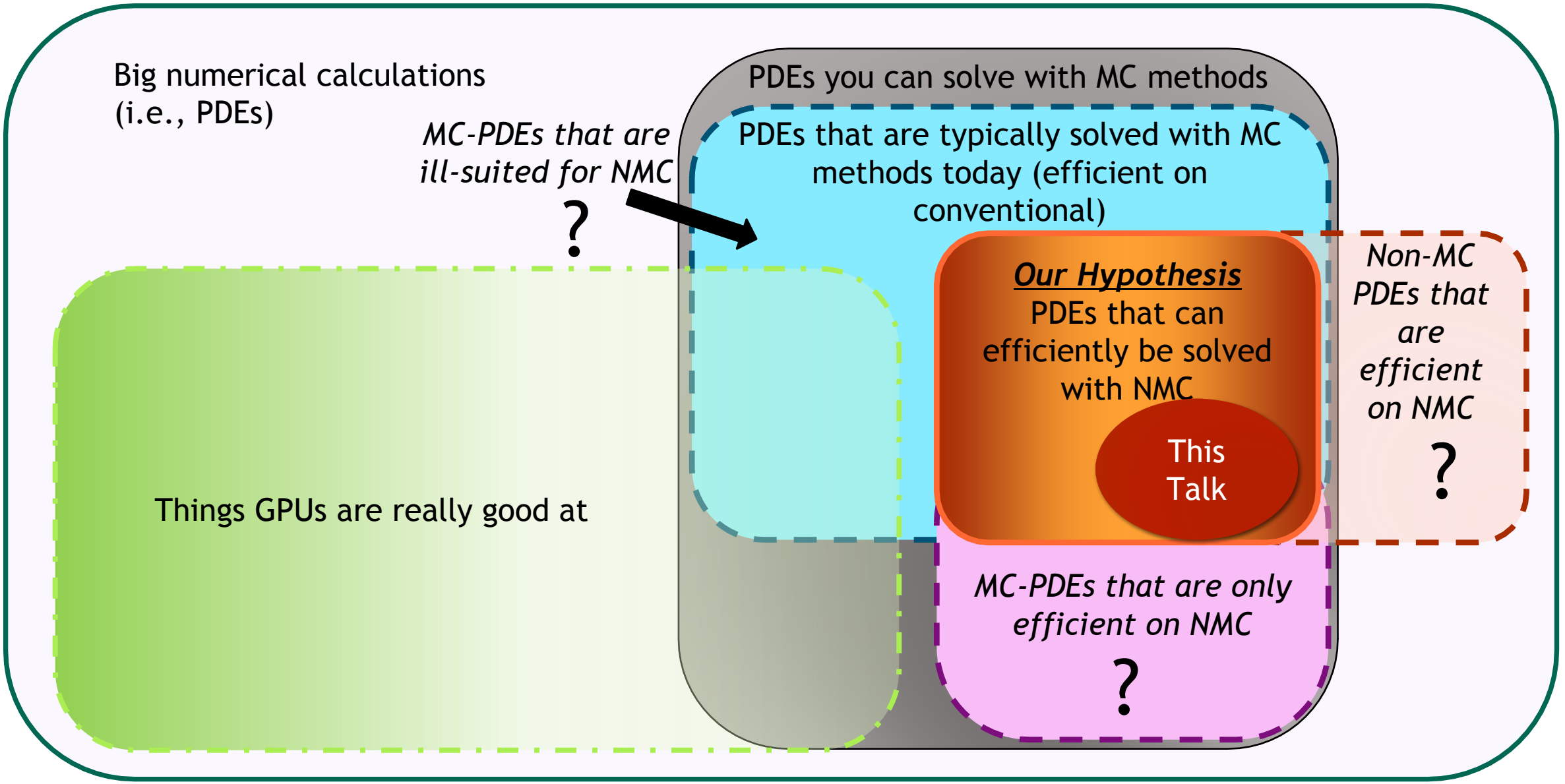
Loihi-specific circuit



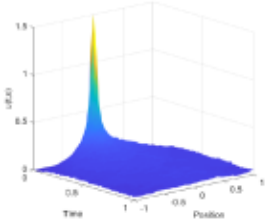
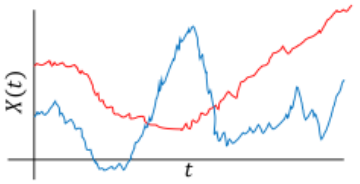
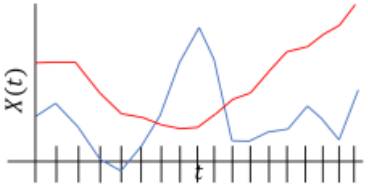
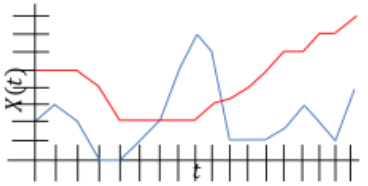
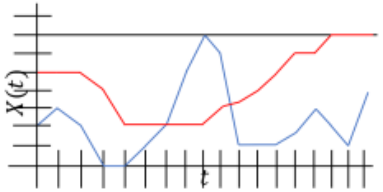
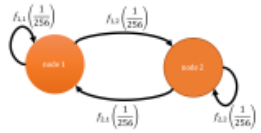
1. Supervisor circuit
   1. Start buffer
   2. Start counter
2. Counter circuit
   1. Buffer neurons
   2. Counter neurons
3. Probabilistic neurons
4. Output neurons



Our hypothesis: There exists a class of scientific computing algorithms for which neuromorphic computing is *efficient*

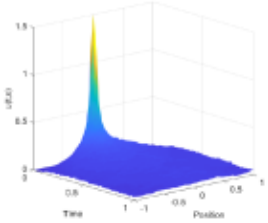
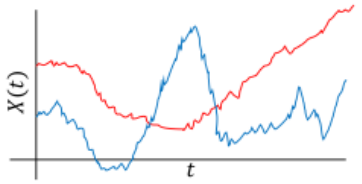
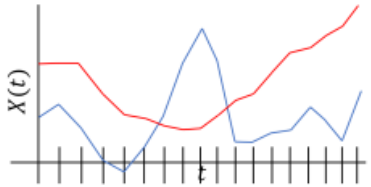


Accuracy Stack for Neuromorphic Implementation

PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Problem	**Approximation**	**Visualization**	**Error/Convergence**
To approximate the expectation, we must sample paths of the stochastic process.	$u(t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(t, X_t^i);\ X_0^{(i)} = x$		$\frac{1}{\sqrt{M}}$
Continuous paths cannot be sampled, we must employ a discretization scheme.	$u(j\Delta t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, X_{j\Delta t}^i);\ X_0^{(i)} = x$		$\sqrt{\Delta t}$
Neurons cannot represent a continuum of locations. Hence we must limit the spatial locations of the walk.	$u(j\Delta t,x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \hat{X}_{j\Delta t}^i);\ \hat{X}_0^{(i)} = x_k$		$\frac{1}{2}j\Delta t\Delta s$
There are a finite number of neurons, so maximum and minimum values for the random walk will exist.	$u(j\Delta t,x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \check{X}_{j\Delta t}^i);\ \check{X}_0^{(i)} = x_k$		varies
Hardware Specific Issues. TrueNorth having quantized probability, for example.	$\mathbb{P} \propto \frac{1}{256}$		varies

Smith et al., in review 2021

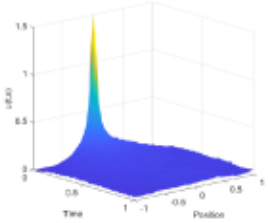
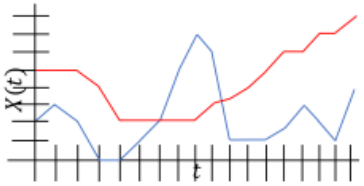
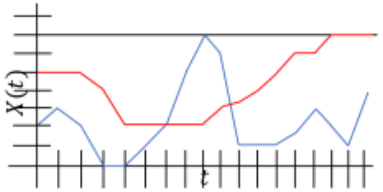
Accuracy Stack for Neuromorphic Implementation

PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Problem	**Approximation**	**Visualization**	**Error/Convergence**
To approximate the expectation, we must sample paths of the stochastic process.	$u(t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(t, X_t^i);\ X_0^{(i)} = x$		$\frac{1}{\sqrt{M}}$
Continuous paths cannot be sampled, we must employ a discretization scheme.	$u(j\Delta t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, X_{j\Delta t}^i);\ X_0^{(i)} = x$		$\sqrt{\Delta t}$

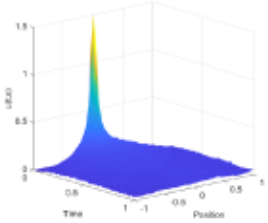
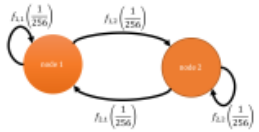
⇧⇧⇧⇧⇧⇧

Present for any implementation

Accuracy Stack for Neuromorphic Implementation

Problem	Approximation	Visualization	Error/Convergence
PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Neuromorphic Specific ⇩ ⇩ ⇩ ⇩ ⇩ ⇩			
Neurons cannot represent a continuum of locations. Hence we must limit the spatial locations of the walk.	$u(j\Delta t, x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \hat{X}^i_{j\Delta t});\ \hat{X}^{(i)}_0 = x_k$		$\frac{1}{2}j\Delta t\Delta s$
There are a finite number of neurons, so maximum and minimum values for the random walk will exist.	$u(j\Delta t, x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \check{X}^i_{j\Delta t});\ \check{X}^{(i)}_0 = x_k$		varies
⇧ ⇧ ⇧ ⇧ ⇧ ⇧			

Accuracy Stack for Neuromorphic Implementation

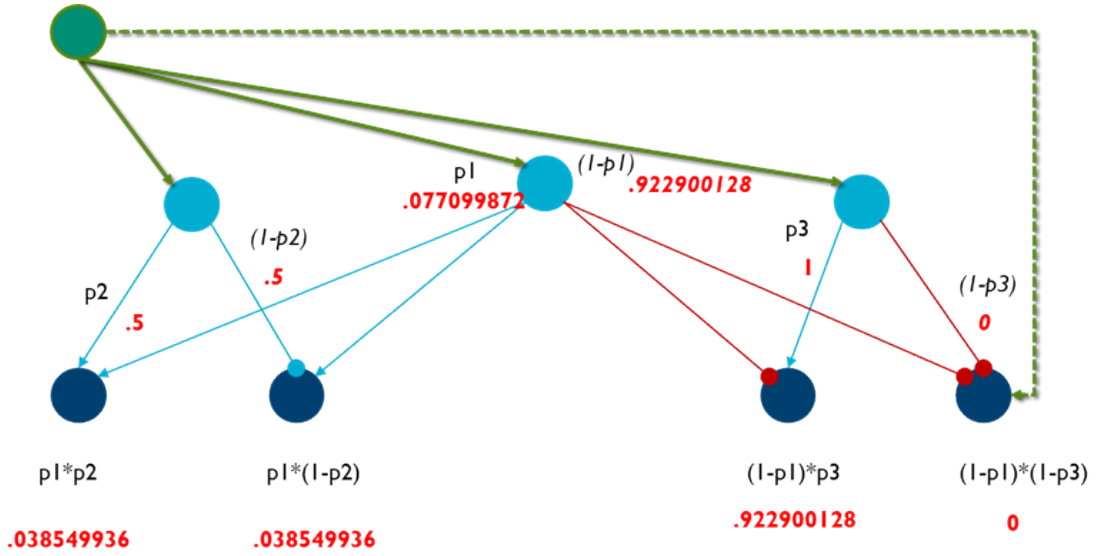
Problem	Approximation	Visualization	Error/Convergence
PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Hardware Specific Issues. TrueNorth having quantized probability, for example.	$\mathbb{P} \propto \frac{1}{256}$		varies *Smith et al., in review 2021*

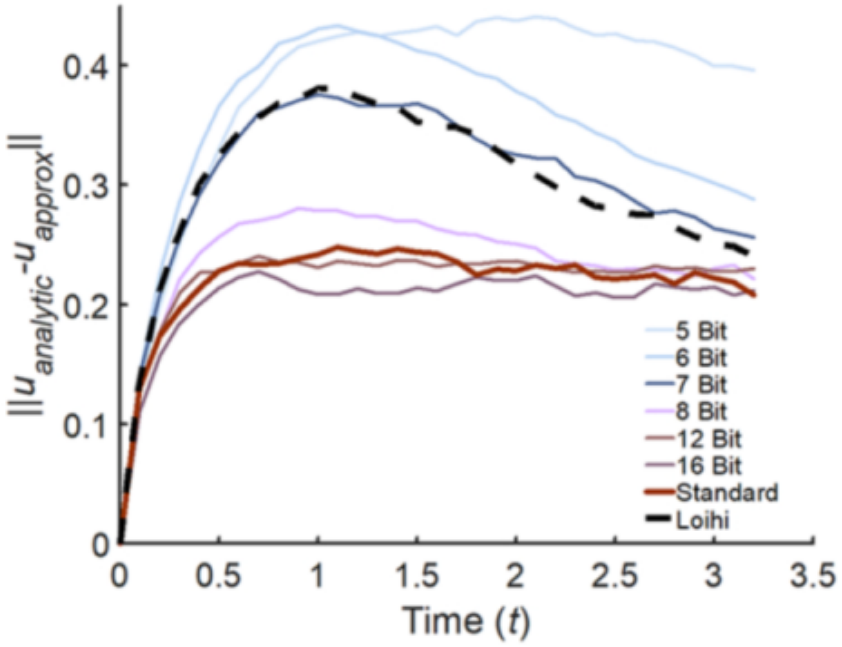
Hardware Specific

⇩ ⇩ ⇩ ⇩ ⇩ ⇩

Precision example: spherical diffusion

- Transition probabilities between neural mesh points are determined by RNGs on probabilistic neurons
- On Loihi, PRNGs are 8-bit, effectively making transition probabilities 8-bit
- Comparing sphere Loihi example to MATLAB simulation with reduced precision suggests Loihi is roughly 7-bit precision





Smith et al., in review 2021

Generating random walks

AR1/Harmonic Well Data Generation:

- A spatial and temporal discretization is chosen, Δx and Δt.
- The SDE is used to determine a discretized update scheme.

$$\mathrm{d}X(t) = -\alpha(X(t) - z)\mathrm{d}t + \sqrt{2D}\mathrm{d}W(t) \;\rightarrow\; X(t_i) = X(t_{i-1}) - \alpha(X(t_{i-1}) - z)\Delta t + \eta_i$$

- A DTMC is constructed using the law of the update scheme and the spatial discretization size.

$$p(x_i \rightarrow x_j) = p\left(X(t_i) \in \left[x_j - \frac{\Delta x}{2}, x_j + \frac{\Delta x}{2}\right] \middle| X(t_{i-1}) = x_j\right)$$

- Collected data yields the number of random walkers on each mesh point at each time step.
- Neuromorphic requires restricting the state space. In this example, $x \in [-L, L]$.