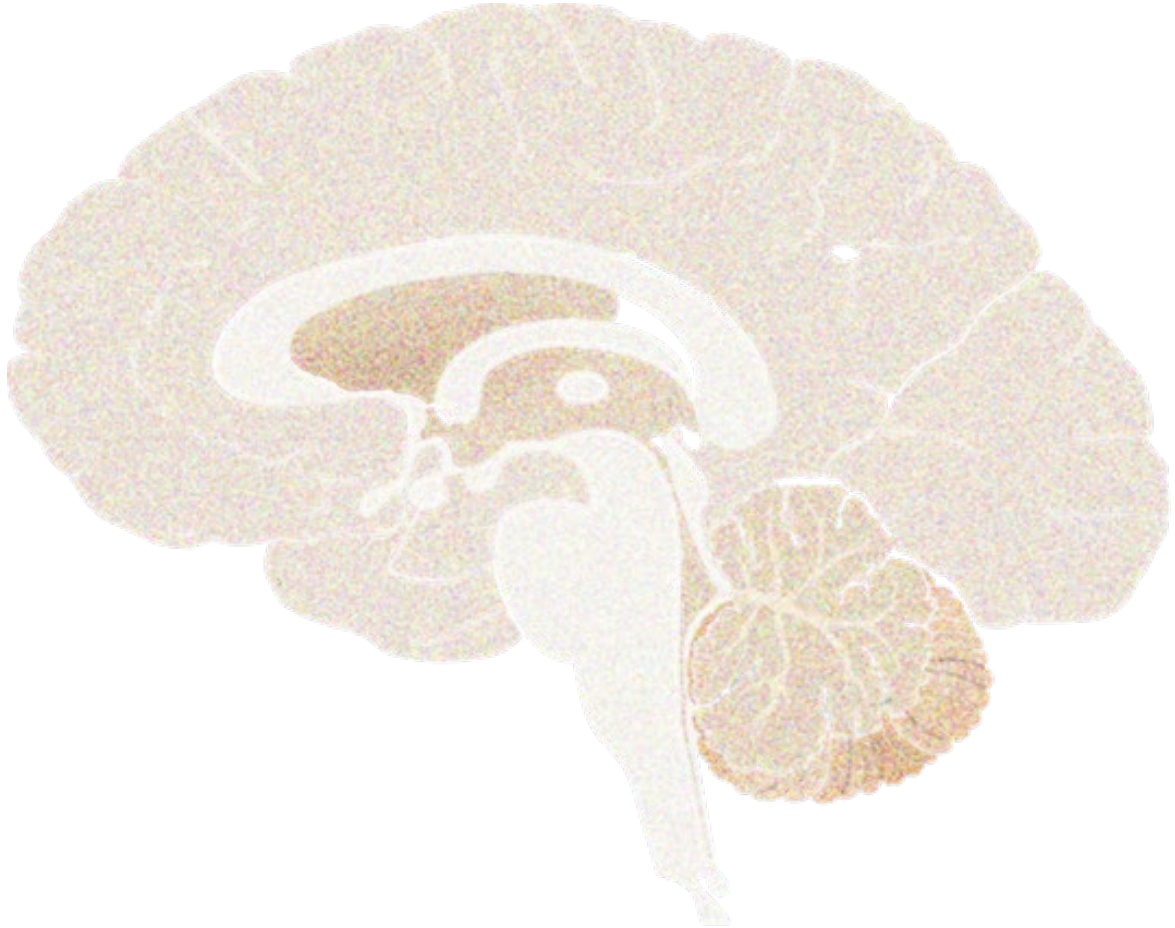
A THEORETICAL FRAMEWORK FOR TIME, SPACE, AND ENERGY SCALING OF NEUROMORPHIC ALGORITHMS

Brad Aimone

Center for Computing Research
jbaimon@sandia.gov



We don’t know how this works. Or for that matter, we don’t even know what it is doing….

Hypothesis: Anything the brain does should be consistent with what the brain is.

The algorithm must match the architecture.

This is our unknown

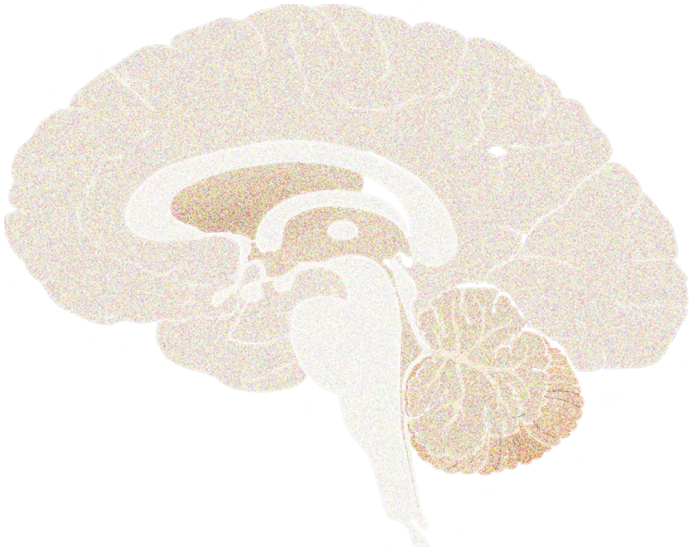
The **algorithm** must match the **architecture**.

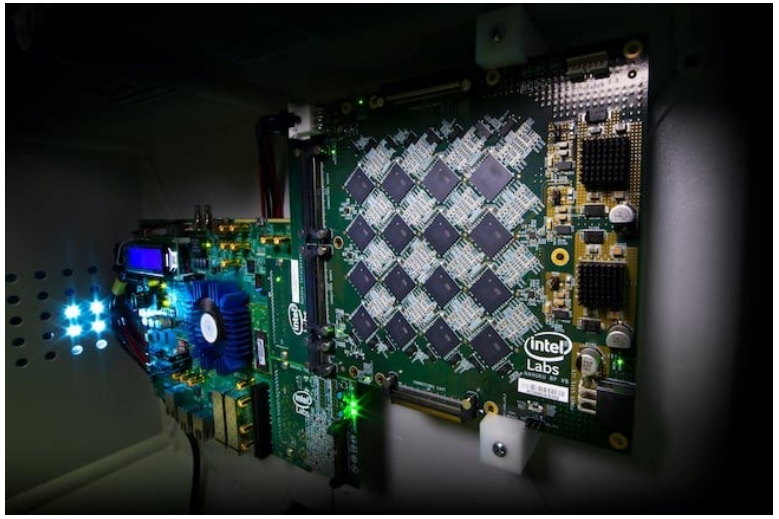
Can we approximate this?

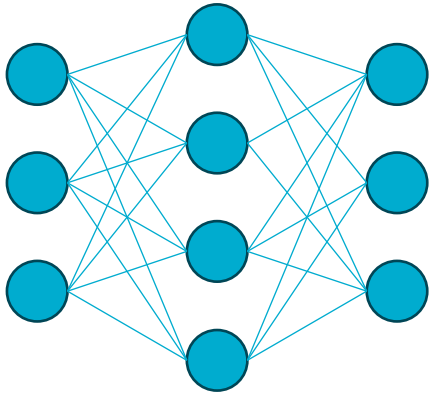
Neuromorphic computing is a potential approximation.

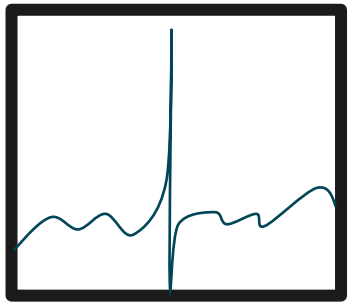
We need to better define what neuromorphic computing is.

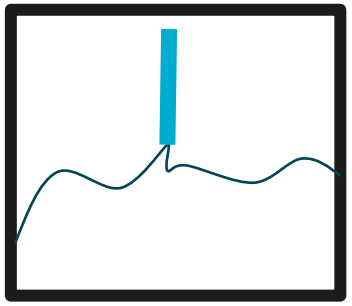
ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

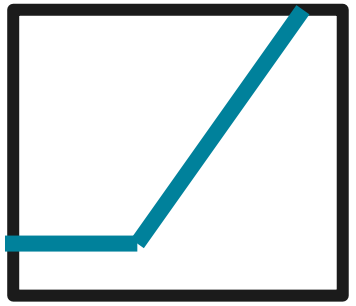






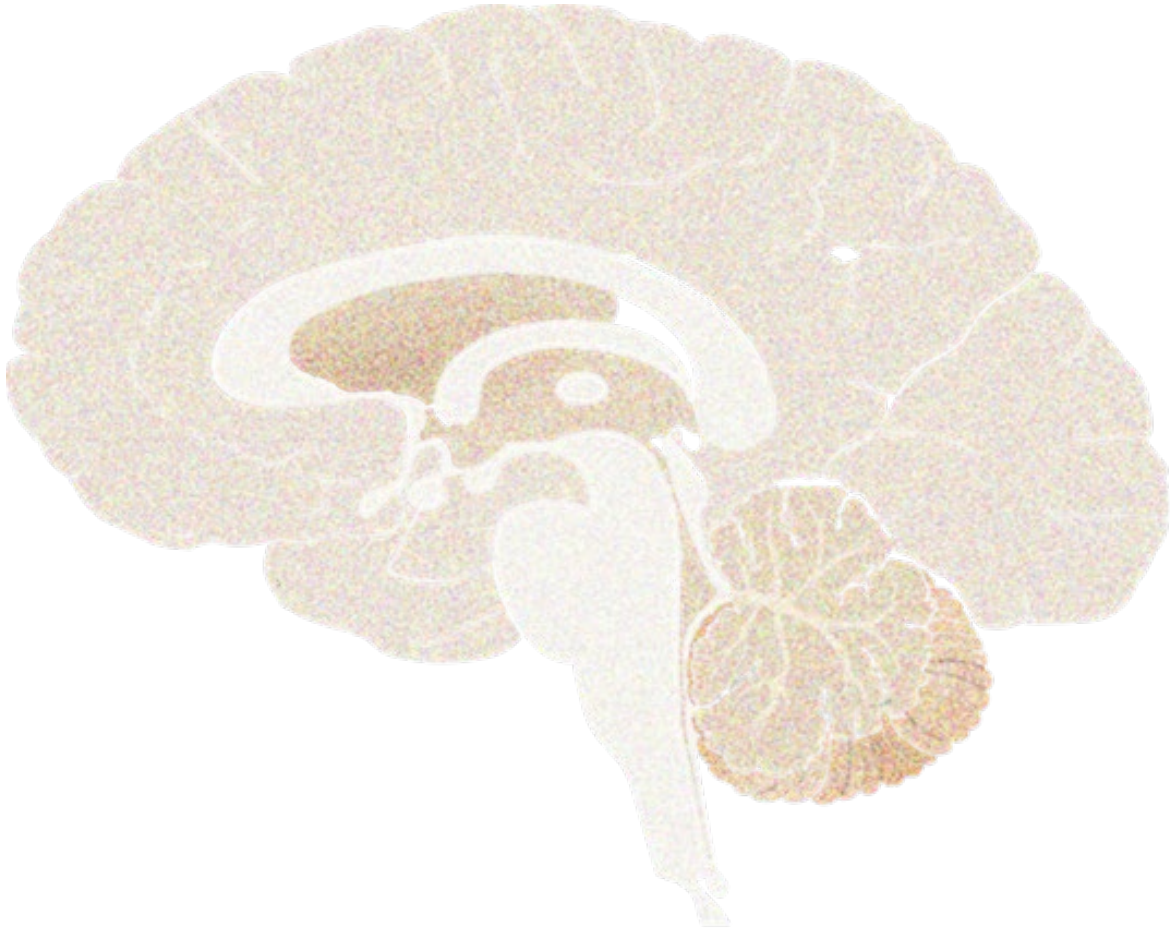






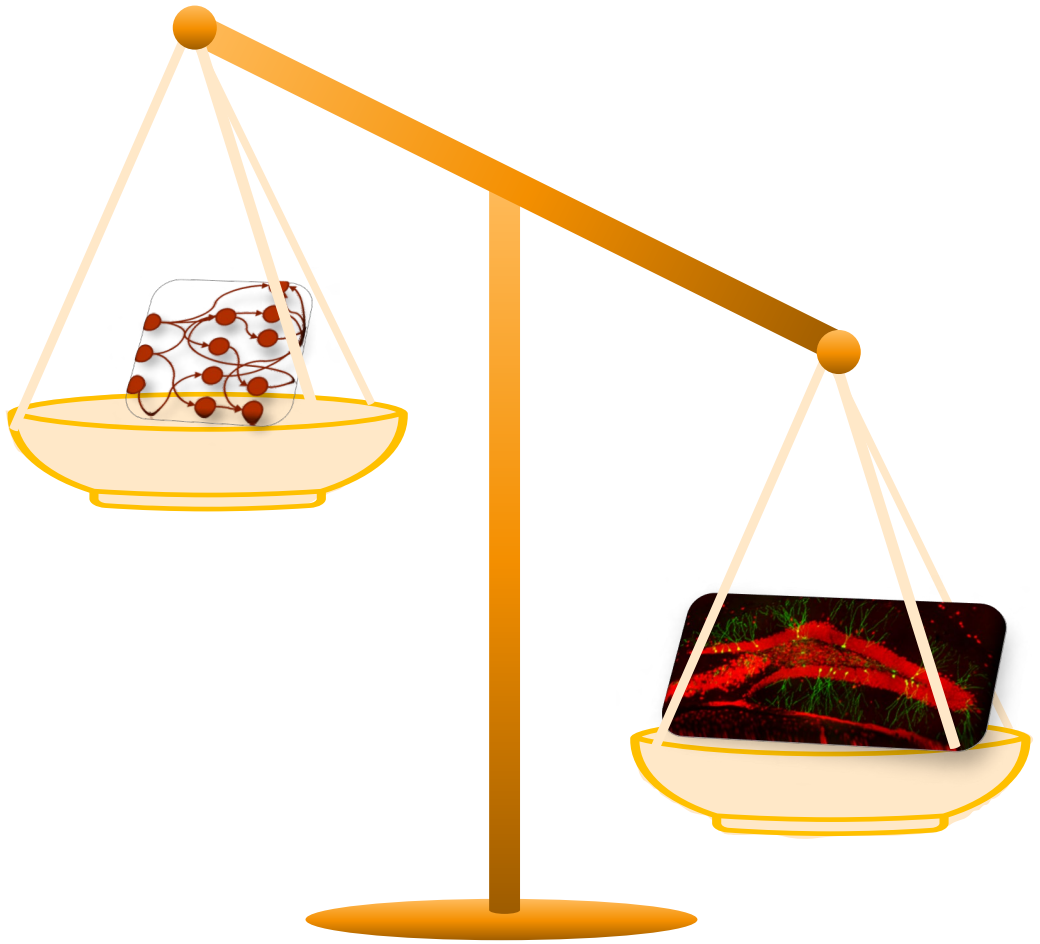
ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓The brain is an existence proof



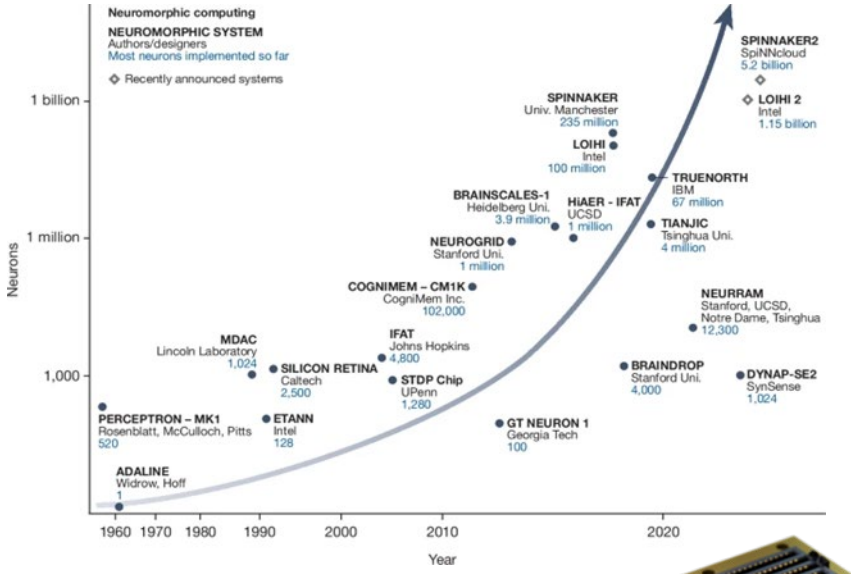
ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓ The brain is an existence proof
- × The brain is hard to understand

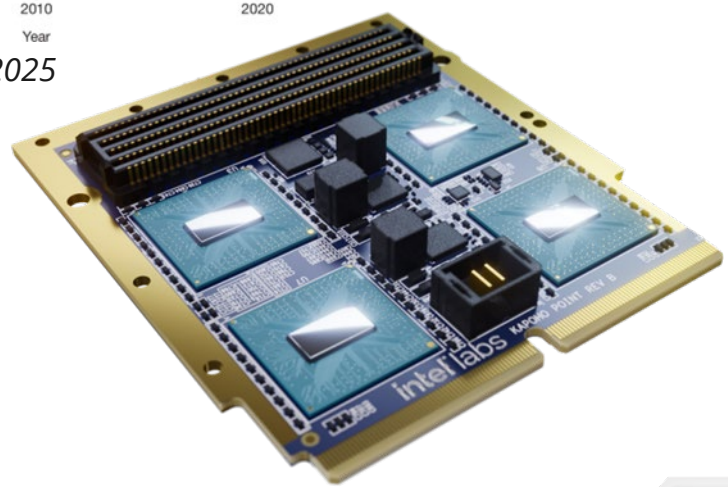


ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓The brain is an existence proof
- ✗The brain is hard to understand
- ✓Scalable hardware exists



Kudithipudi et al, Nature 2025

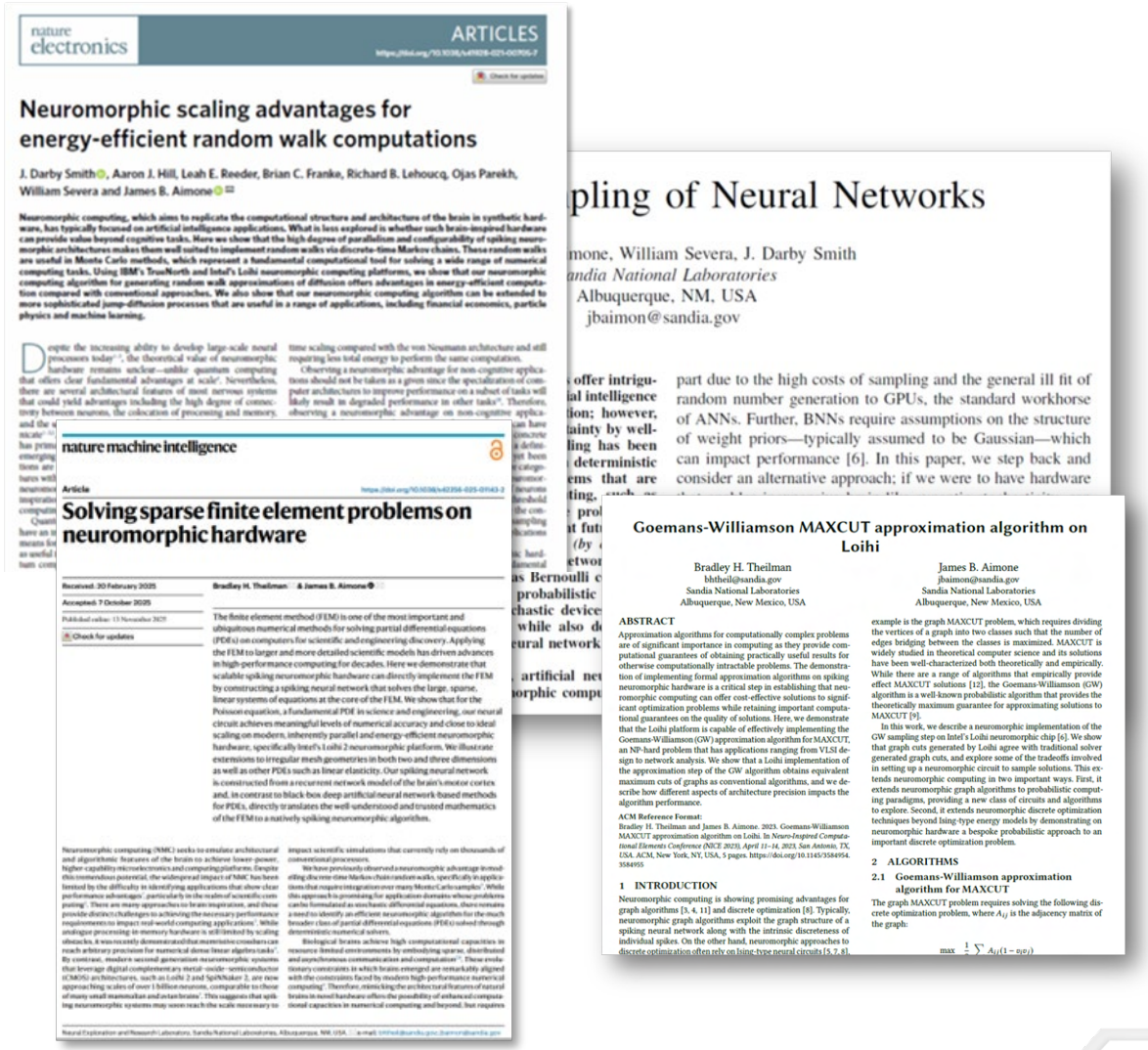


ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓ The brain is an existence proof
- ✗ The brain is hard to understand
- ✓ Scalable hardware exists
- ✗ People don’t really know how to program it

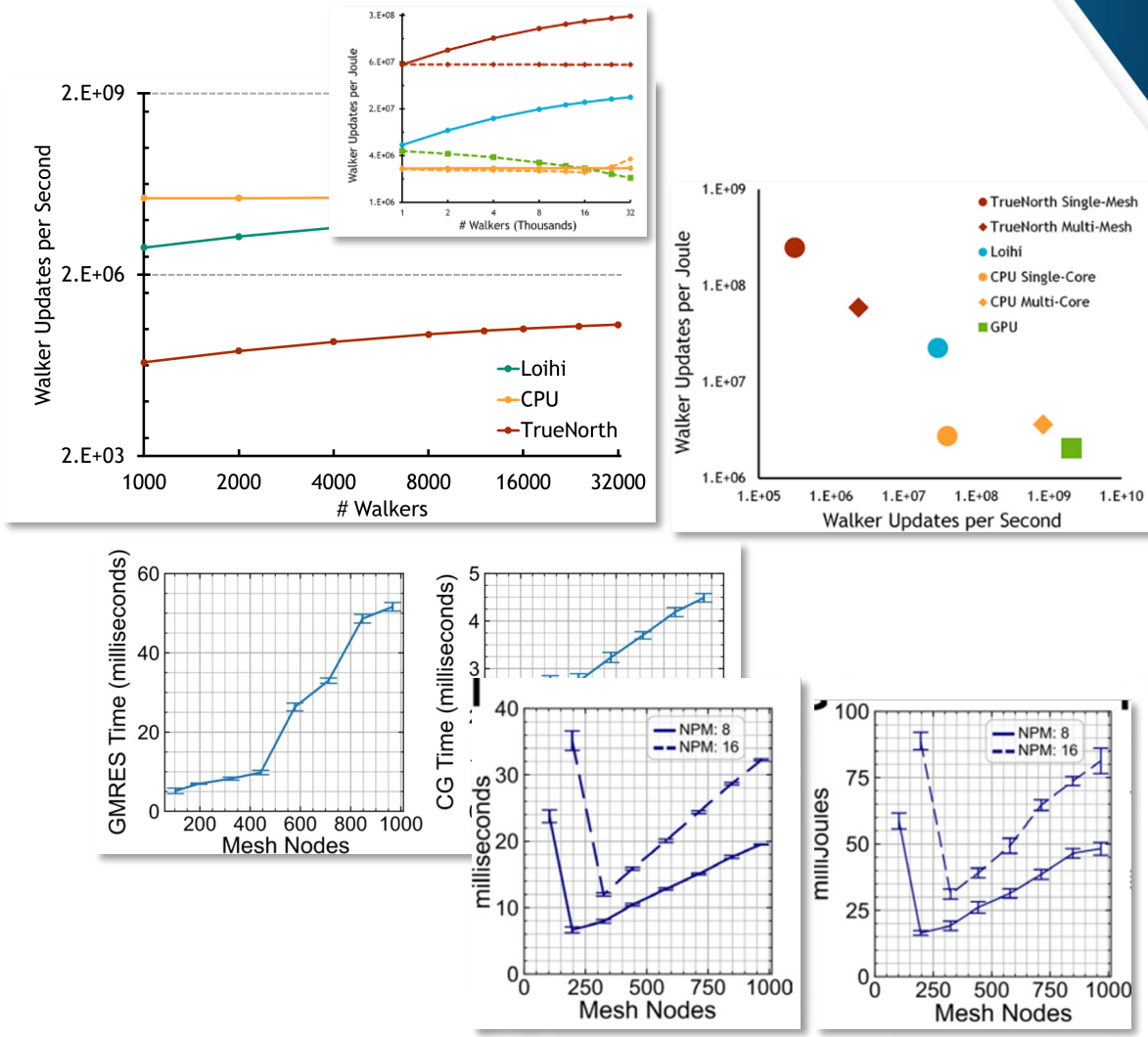
ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓ The brain is an existence proof
- ✗ The brain is hard to understand
- ✓ Scalable hardware exists
- ✗ People don't really know how to program it
- ✓ There are algorithm successes



ARE WE READY FOR A COMMON FRAMEWORK FOR ANY NEURAL COMPUTATION?

- ✓ The brain is an existence proof
- ✗ The brain is hard to understand
- ✓ Scalable hardware exists
- ✗ People don't really know how to program it
- ✓ There are algorithm successes
- ✗ Results are largely heuristic and empirical



A START TOWARDS A THEORETICAL FRAMEWORK FOR NEUROMORPHIC COMPUTING

- *Challenge*: Neuromorphic computing has been hardware motivated. As such, the field lacks the theoretical foundation that conventional and quantum computing have.
  - There is no widely agreed upon "model of neural computation"
  - There is no standard method for formal analysis of neuromorphic algorithms, architectures, etc.

A Theoretical Framework for Time, Space, and Energy Scaling of Neuromorphic Algorithms

James B Aimone
Neural Exploration & Research Laboratory
Sandia National Laboratories
Albuquerque, NM 87185
jbaimon@sandia.gov

Revised March 2026

Abstract

Neuromorphic computing (NMC) is increasingly viewed as a low-power alternative to conventional von Neumann architectures such as central processing units (CPUs) and graphics processing units (GPUs), however the computational value proposition has been difficult to define precisely.

Here, we propose a computational framework for analyzing NMC algorithms and architectures. Using this framework, we demonstrate that NMC can be analyzed as general-purpose and programmable even though it differs considerably from a conventional stored-program architecture. We show that the time and space scaling of idealized NMC has comparable time and footprint tradeoffs that align with that of a theoretically infinite processor conventional system. In contrast, energy scaling for NMC is significantly different than conventional systems, as NMC energy costs are event-driven. Using this framework, we show that while energy in conventional systems is largely determined by the scheduled operations determined by the structural algorithm graph, the energy of neuromorphic systems scales with the activity of the algorithm, that is the activity trace of the algorithm graph. Without making strong assumptions on NMC or conventional costs, we demonstrate which neuromorphic algorithm formulations can exhibit asymptotically improved energy scaling when activity is sparse and decaying over time. We further use these results to identify which broad algorithm families are more or less suitable for NMC approaches.

1 Introduction

Neuromorphic computing ('NMC') approaches to computation have been proposed for many years [1–3], and today NMC hardware is increasingly available and can be implemented at scales approaching 10^9 neurons in silicon [4], although still these systems remain far from the brain's complexity [5]. One implication of the success of implementing large-scale NMC systems is that it is increasingly evident that the biggest challenge facing the neuromorphic field, at least in the near-term, is the identification of which applications are well suited for its use. While scalable NMC platforms were originally motivated, in part, by large-scale brain simulations [6–9], the growing need for low-power solutions at both the edge and in high-performance computing systems has increased interest in NMC to address energy challenges in computation broadly.

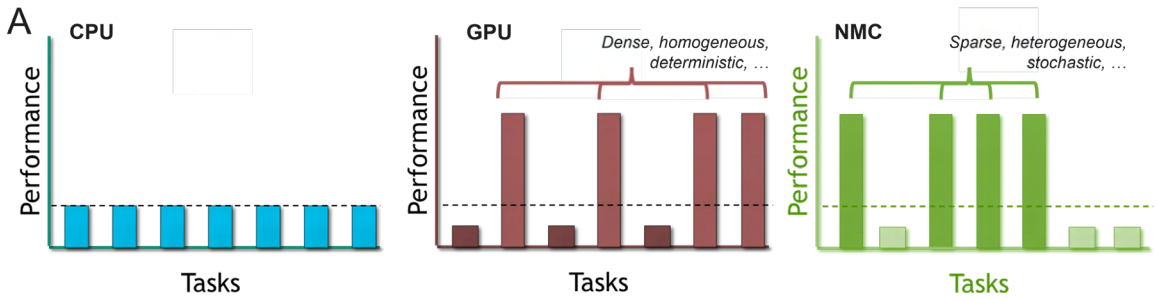
Key to their value proposition is that today's digital NMC systems are generically programmable—aside from some constraints of fan-in/fan-out, these systems can implement arbitrary computational graphs. Conceptually, this compatibility includes any algorithm that can be formulated as a threshold gate (TG) circuit, artificial neural network (ANNs), or any other more complex ensemble of neurons. As the ability to implement TGs confers Turing completeness [10], and ANNs confer universal function approximation [11], we can deduce that NMC is both universal in terms of algorithm compatibility and its ability to approximate functions.

Given this universality, the relevant question of NMC is not "*can* neuromorphic solve this task?", but rather "*should* NMC be used to solve this task?". While NMC is by definition general purpose in potential, based on the 'No Free Lunch' theorem as applied to computer architectures, it should be expected that it will

1

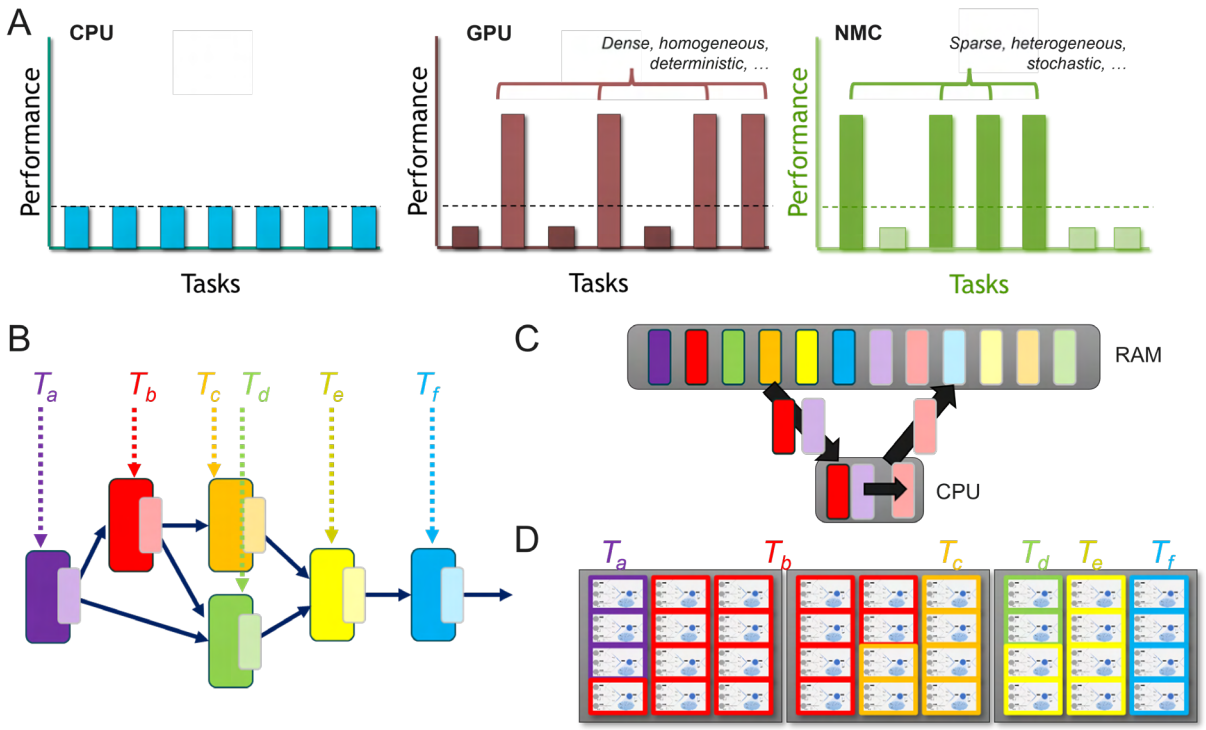
HOW DOES NMC RELATE TO GPUS AND OTHER VON NEUMANN ARCHITECTURES?

- Expectation that neuromorphic systems will excel at some tasks but be worse at others
  - “Specialized general purpose”, like GPUs but specialized to different tasks
  - “No Free Lunch”



HOW DOES NMC RELATE TO GPUS AND OTHER VON NEUMANN ARCHITECTURES?

- Expectation that neuromorphic systems will excel at some tasks but be worse at others
  - “Specialized general purpose”, like GPUs but specialized to different tasks
  - “No Free Lunch”
- Neuromorphic is fully *in memory computing*
  - No stored program
  - Allows event-driven, asynchronous operation



OUTLINE

1. Formal definitions

2. First crack at a complexity analysis

3. Algorithm successes

OUTLINE

1. Formal definitions
2. First crack at a complexity analysis
3. Algorithm successes

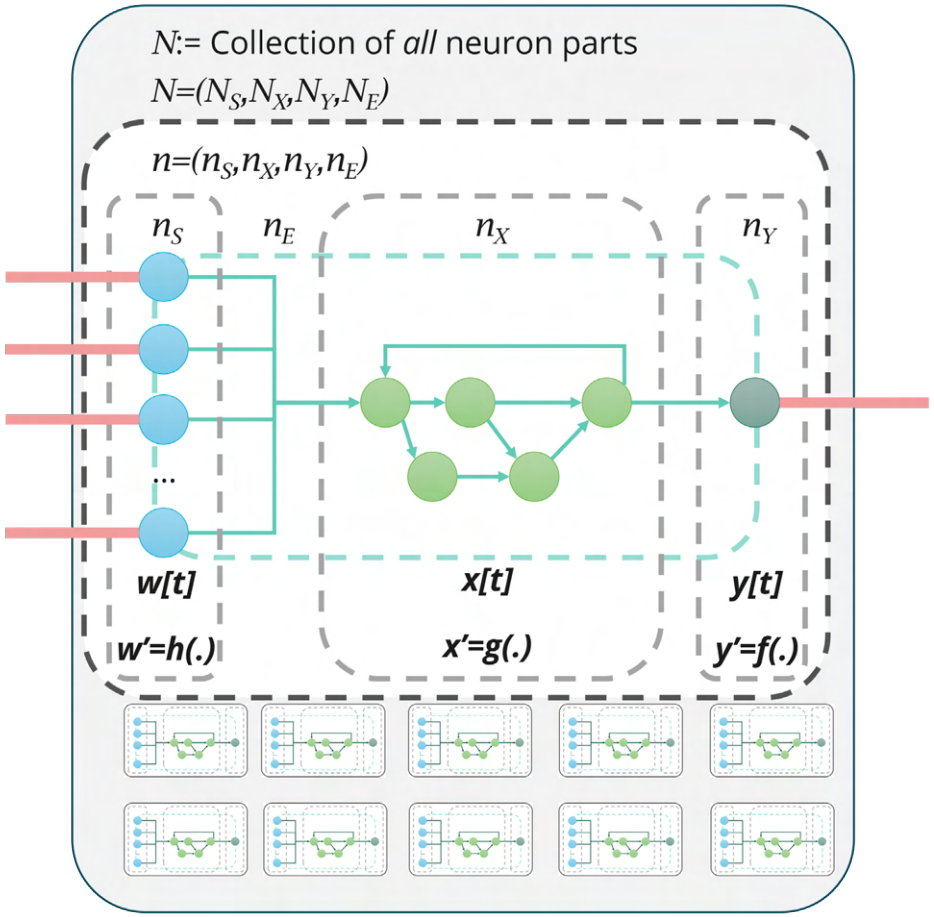
FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

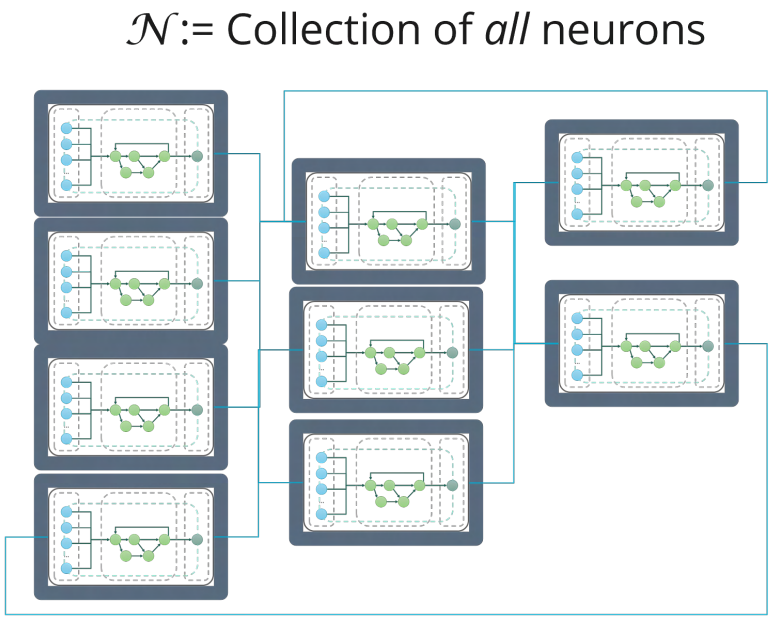
Definition (Neuron). A neuron is a compute element represented as a (potentially cyclic)directed subgraph $n = (n_S, n_X , n_Y , n_E)$, where:

- n_S: Synaptic input nodes, representing incoming spiking signals from other neurons (s[t])and exhibiting synaptic dynamics related to the representation (w[t]), and evolution ($h(\cdot)$)of synaptic weight, synaptic delay, stochasticity, and synaptic plasticity.
- n_X: Internal compute nodes describing the neuron's internal state variables (x[t]) and the neuron's internal state dynamics ($g(\cdot)$).
- n_Y: Output nodes, representing the neuron's observable output (y[t]) and the generation of that output ($f(\cdot)$). For spiking neurons, $y[t] \in \{0, 1\}$.
- n_E: Internal edges, representing directed connections between n_S, n_X , and n_Y. The internal edges n_E may contain self-loops and directed cycles to represent intrinsic dynamics (e.g., decay, refractoriness, or adaptation).

FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

Definition S1.1 (Neuron). A neuron is a compute element represented as a (potentially cyclic)directed subgraph $n = (n_S, n_X, n_Y, n_E)$, where:





FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

Definition (Synapse). Let *N* represent the set of isolated neuron subgraphs in a neuromorphic system, where each synapse is a directed synaptic communication channel from a source neuron $n_i \in N$ to a target neuron $n_j \in N$. We take the set of synaptic communication channels to be a set of directed edges

$$S \subseteq N_Y \times N_S$$

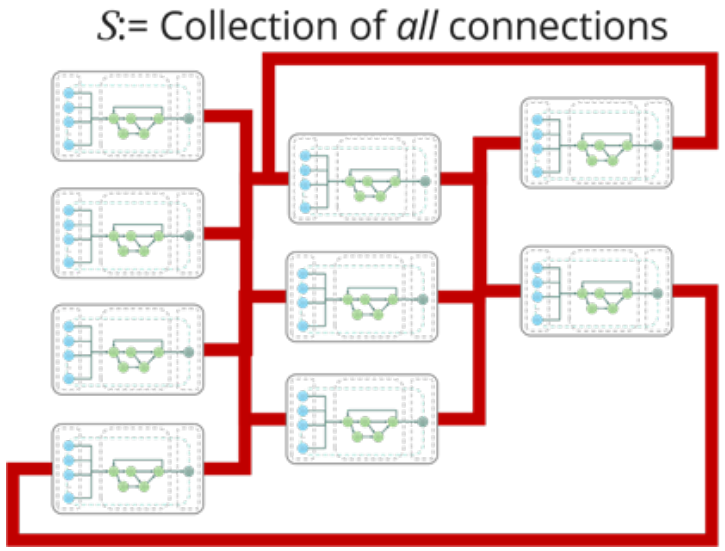
Each synapse connects an output node $n_{yi} \in n_{i,Y}$ of the source neuron n_i to a synaptic input node $n_{si}\ j \in n_{j,S}$ of the target neuron n_j via a directed edge $s_{ij} \in S$, where:

- n_{yi} : The output node of the source neuron n_i, representing the observable spiking output $y_i[t] \in \{0, 1\}$.
- n_{sij}: The synaptic input node of the target neuron n_j, representing post-synaptic state associated with the connection, including synaptic weight $w_{ij}[t]$ and any synaptic dynamics (e.g., delay, stochasticity, filtering, and plasticity).
- s_{ij}: The directed edge connecting n_{yi} to n_{sij}, representing discrete spike-event transmission between neurons, $s_{ij}[t] \in \{0, 1\}$.

FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

Definition S1.2 (Synapse). Let N represent the set of isolated neuron subgraphs in a neuromorphic system, where each synapse is a directed synaptic communication channel from a source neuron $n_i \in N$ to a target neuron $n_j \in N$. We take the set of synaptic communication channels to be a set of directed edges

$$S \subseteq N_Y \times N_S$$



FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

Definition (Neuromorphic Computing Architecture). A neuromorphic computing architecture is a distributed computing system capable of representing a set of neurons N and a set of synaptic communication channels S as a directed graph $G_{N,\text{arch}} = (N_{\text{arch}}, S_{\text{arch}})$, where:

- N_{arch}: The union of neuron components of the set of isolated neuron subgraphs N , where each neuron $n \in N$ is represented as $n = (n_S, n_X , n_Y , n_E)$, as defined in Definition S1.1.
- S_{arch}: The set of directed synaptic communication channels between neuron subgraphs, where each $s_{i\,j} \in S_{\text{arch}}$ connects a source output node n_{yi} to a target synaptic input node n_{sij}, as defined in Definition S1.2

FORMAL DEFINITION OF NEURONS, SYNAPSES, NEURAL ARCHITECTURES AND NEURAL ALGORITHMS

Definition (Neuromorphic Algorithm). A neuromorphic algorithm is an algorithm implemented on a neuromorphic computing architecture $G_{N,\text{arch}} = (N_{\text{arch}}, S_{\text{arch}})$. The algorithm is specified by (i) a selection of an algorithm subgraph $G_{N,\text{alg}} = (N_{\text{alg}}, S_{\text{alg}})$ and (ii) a parameterization of the selected neuron-local state and update rules, where:

- N_{alg}: The combined graph of selected neuron subgraphs, representing the computational nodes of the algorithm.
- S_{alg}: The selected set of synaptic communication channels, representing the directed edges between distinct neurons. The algorithm graph $G_{N,\text{alg}} = (N_{\text{alg}}, S_{\text{alg}})$ represents the ideal computational graph for the algorithm, with $N_{\text{alg}} \subseteq N_{\text{arch}}$, $S_{\text{alg}} \subseteq S_{\text{arch}}$.

UPSHOT OF FRAMEWORK DEFINITIONS

- Clarifies neuromorphic algorithm [$G_N = (N,S)$] resource count
  - *N* (neurons) captures all computation
    - Neurons can have arbitrary complexity, but this is likely a constant cost
    - *Opens up neuron complexity vs algorithm scaling direction*
  - *S* (synapses) captures the non-local communication
    - If fan-out/fan-in is bounded, then synapse count scales no worse than neuron count
    - *Allows us to isolate distinct energy costs*

OUTLINE

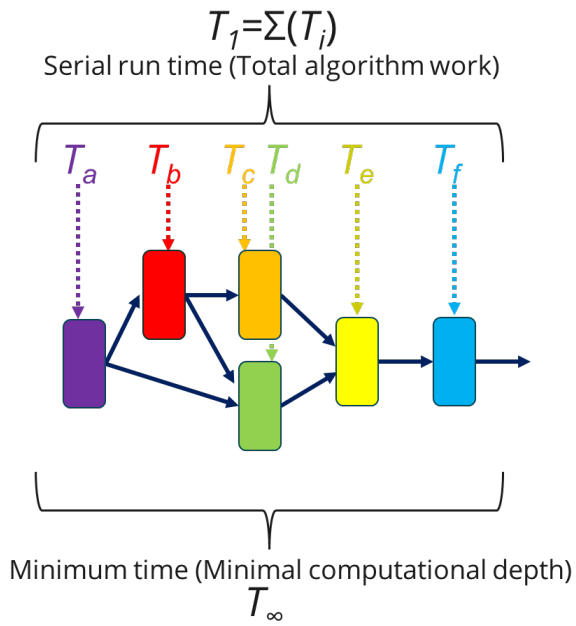
1. Formal definitions

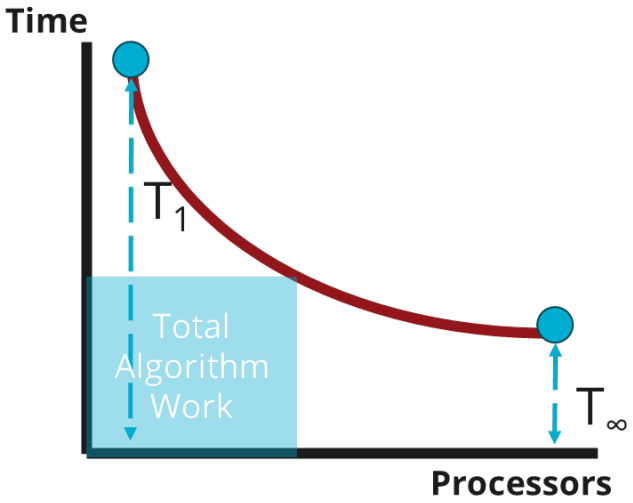
2. First crack at a complexity analysis

3. Algorithm successes

NEUROMORPHIC ALGORITHM TIME & SPACE COSTS

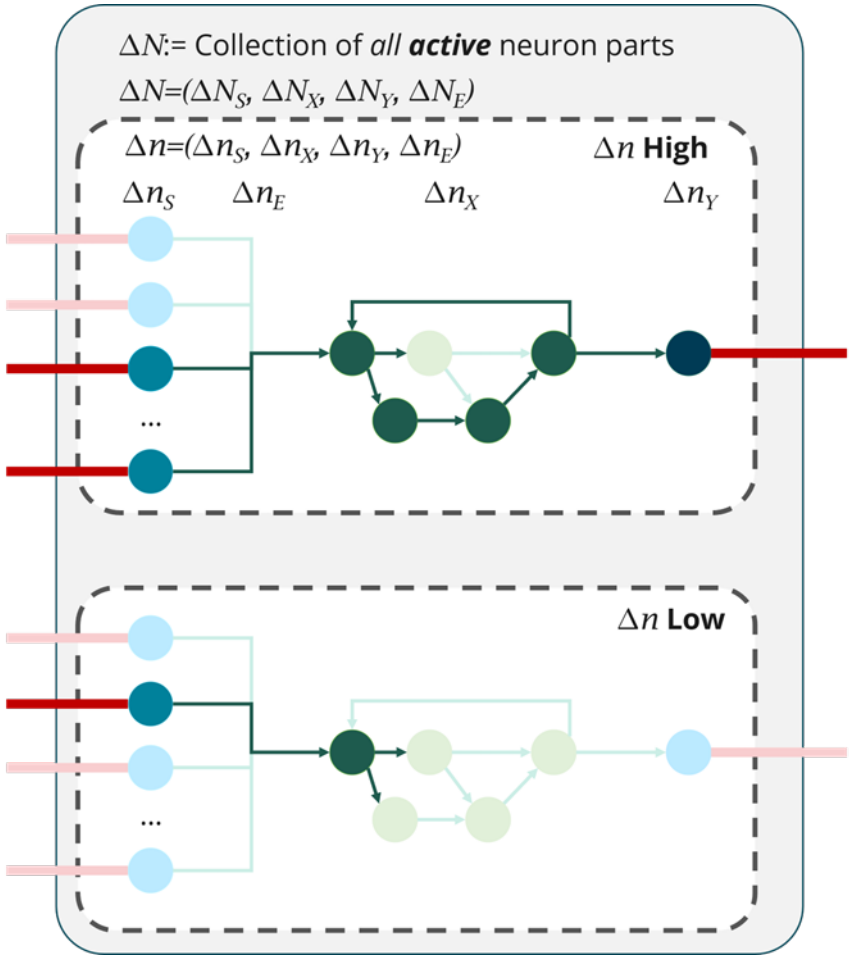
- Ideal NMC executes at parallel depth limits
  - Runtime is governed by the critical path of the time-unrolled graph, i.e. $C_T \sim T_\infty\left(G_N^{(T)}\right)$.
  - *Brent's Theorem applies to neuromorphic algorithms*
- No generic asymptotic time/space advantage
  - NMC aligns with classical parallel depth/work tradeoffs rather than changing complexity class.
- The tradeoff shifts to instantiated footprint
  - Because the "program is the circuit," footprint scales with deployed graph size.





NEUROMORPHIC ENERGY COSTS REQUIRE A TRACE-BASED VIEW

- Structure is not enough for energy
  - The instantiated graph $G_N = (N, S)$ tells us what can happen, not what actually happens.
- Event-driven systems pay *only* for realized activity
  - Rnergy depends on which neuron updates and synaptic events occur, i.e. on $\boldsymbol{\Delta G_N(t) = (\Delta N, \Delta S)}$.
  - Energy use is directly related to physical changes in in-memory hardware (flipping a bit, charging a wire)
- This is less convenient a priori, but unavoidable
  - For neuromorphic (and biological) computation, energy cannot in general be inferred from structure alone.



NEUROMORPHIC ALGORITHM ENERGY COSTS

- Conventional energy tracks scheduled work
  - Operations and memory accesses give energy proportional to executed work, up to hierarchy-dependent constants.
- Neuromorphic energy tracks cumulative activity
  - $C_E(G_N; T) \propto \Sigma_t |\Delta N(t)| + \Sigma_t |\Delta S(t)|$
  - $C_E(G_N; T) = O(\sum_{t=0}^{T-1} |\Delta S(t)|)$
- Advantage is conditional, not universal
  - If activity is dense, no gain; if traces are sparse/decaying, energy can scale with realized activity rather than worst-case structure.

OUTLINE

1. Formal definitions

2. First crack at a complexity analysis

3. Algorithm successes

EVALUATING NEUROMORPHIC ALGORITHMS WITH FRAMEWORK

Structural Metrics

- Graph homogeneity
  - $\mathrm{H}(t) \coloneqq \max_{op} |v \in V_t : \mathrm{type}(v) = op|$
- Structural reuse factor
  - $R_T(G_N) = \frac{T_1\left(G_N^{(T)}\right)}{|G_N|}$
- Fan-out
  - $\mathrm{K}(G_N) \coloneqq \max_{n_y \in N_Y} \deg_S^+(n_y)$
  - $\overline{K}(G_N) \coloneqq \frac{|S|}{|N_Y|}$

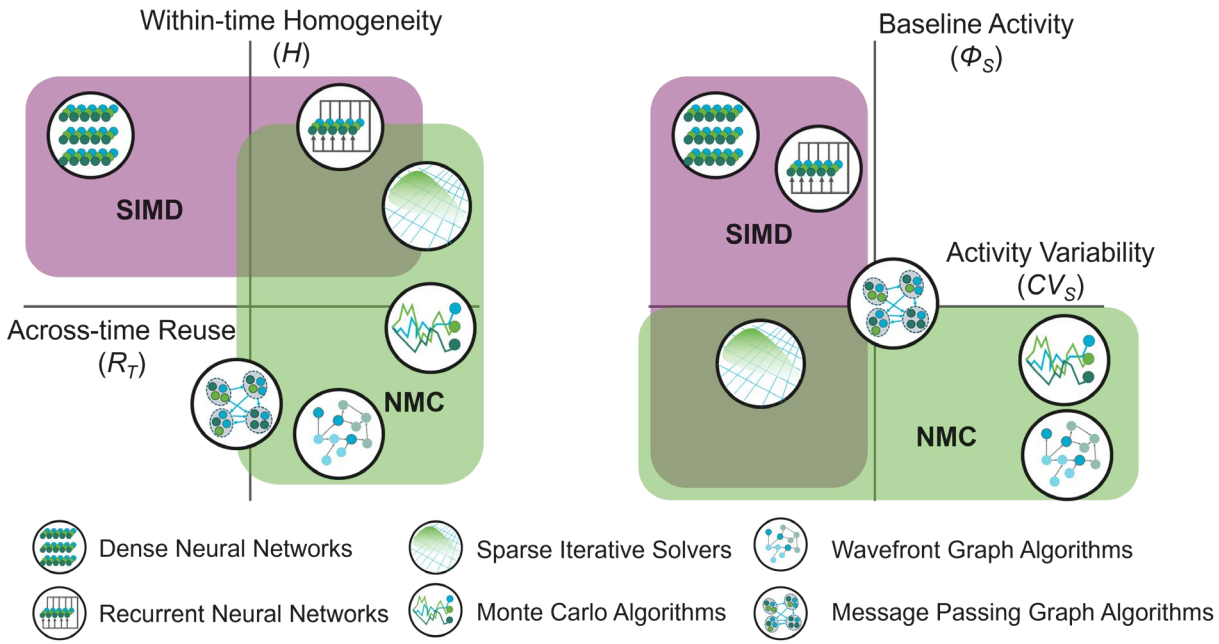
Trace Metrics

- Synaptic activity / sparsity
  - $\Phi_S(G_N; T) = \frac{1}{|S|T} \sum_{t=0}^{T-1} |\Delta S(t)|$
- Activity decay / growth
  - $\Lambda_S(G_N; T) = \frac{1}{T-1} \sum_{t=0}^{T-2} \left(a_S(t+1) - a_S(t)\right)$
- Activity variability
  - $\mathrm{C}V_S(G_N; T) = \frac{\sqrt{\mathrm{Var}(A_S)}}{E[A_S]}$

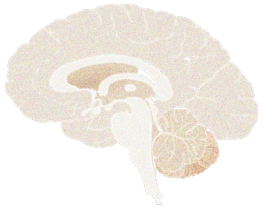
$$A_S \coloneqq \sum_{t=0}^{T-1} |\Delta S(t)| \qquad a_S(t) \coloneqq |\Delta S(t)|$$

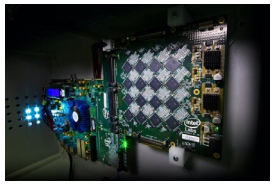
UPSHOT OF THIS FRAMEWORK

- Neuromorphic advantage ***in energy***
  - Formally simply another extreme parallel architecture, but perhaps can realize near-ideal parallel scaling
  - Energy advantage is conditional on sparse activity (ΔG low)
  - Empirical demonstration on DTMC random walks and finite element simulations.
- Advantage relative to GPUs
  - High use of temporal recurrence (allows amortization of spatial cost)
  - High heterogeneity, thread divergence, stochasticity, etc.

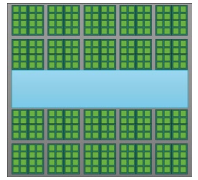


The Brain / Neuromorphic





GPUs / SIMD Architectures



Structural Metrics

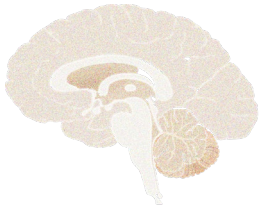
Structural Heterogeneity	Structural Homogeneity
Recurrence / Structural Reuse	Feed-forward / Acyclic
Local Connectivity	Global Connectivity

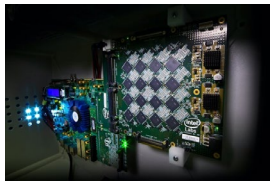
Trace Metrics

Sparse Activity	Dense Activity
Time-dependent Activity	Time-homogeneous Activity
• Thread Divergence	Thread Convergence
• Stochasticity okay	Stochasticity challenging

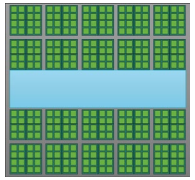
ARTIFICIAL FEED FORWARD NEURAL NETWORKS

The Brain / Neuromorphic





GPUs / SIMD Architectures



Structural Metrics

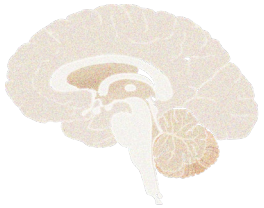
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

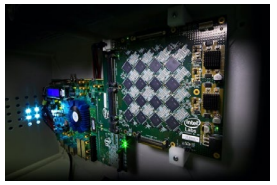
Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

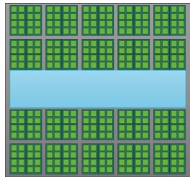
GRAPH ANALYTICS (MESSAGE PASSING, ETC)

The Brain / Neuromorphic





GPUs / SIMD Architectures



Structural Metrics

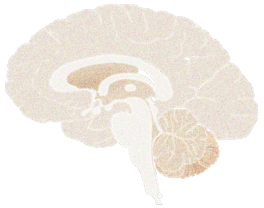
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

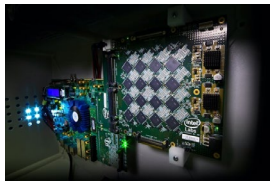
Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

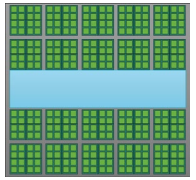
SPARSE LINEAR SOLVERS (CONJUGATE GRADIENT, ETC)

The Brain / Neuromorphic

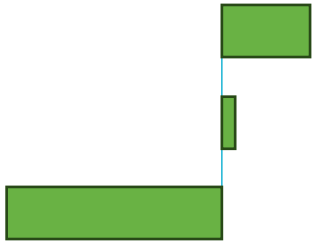




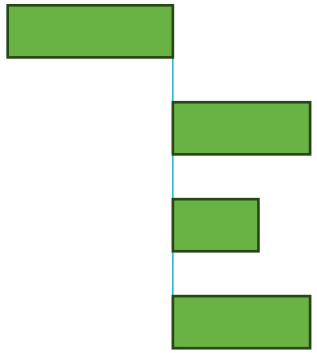
GPUs / SIMD Architectures



Structural Metrics

Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

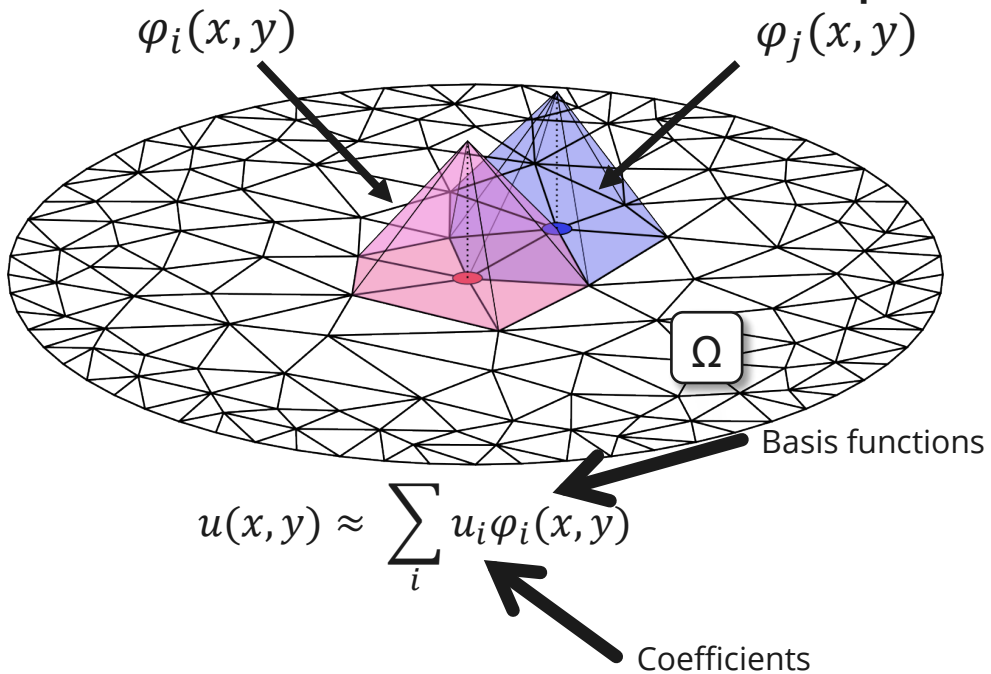
Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

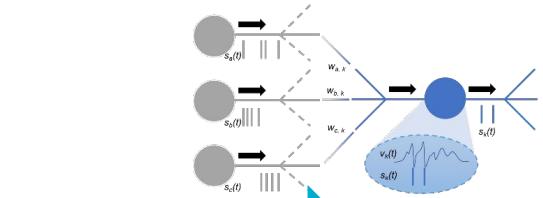
CAN WE TACKLE FEM WITH PROBABILISTIC NEURAL HARDWARE?

$$\nabla^2 u = f$$

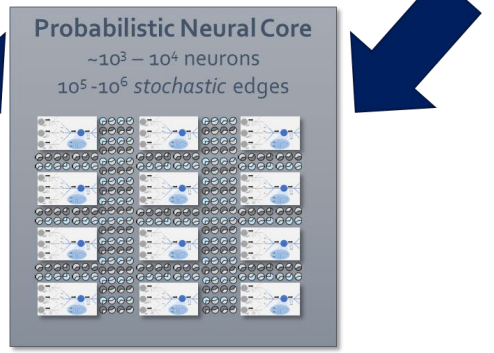
Finite Element Methods – Gold Standard but expensive

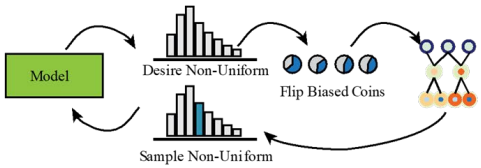


Sparse, linear system $A\vec{u} = \vec{b}$



?????



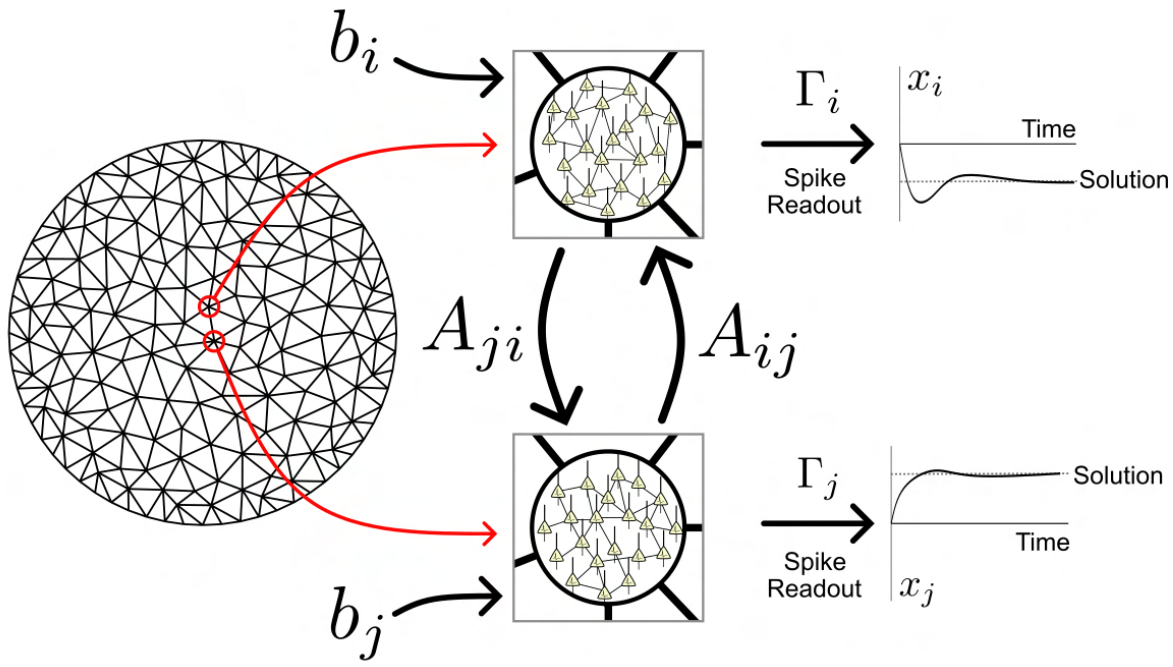


NEUROMORPHIC **FINITE ELEMENT METHODS?**

Poisson Equation

Sparse Linear System

$$\nabla^2 u = f \xrightarrow[\text{Discretization}]{} A\vec{x} = \vec{b}$$



Neuromorphic virtues:

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

nature machine intelligence

Article

Solving sparse finite element problems on neuromorphic hardware

Received: 20 February 2025

Accepted: 7 October 2025

Published online: 13 November 2025

Check for updates

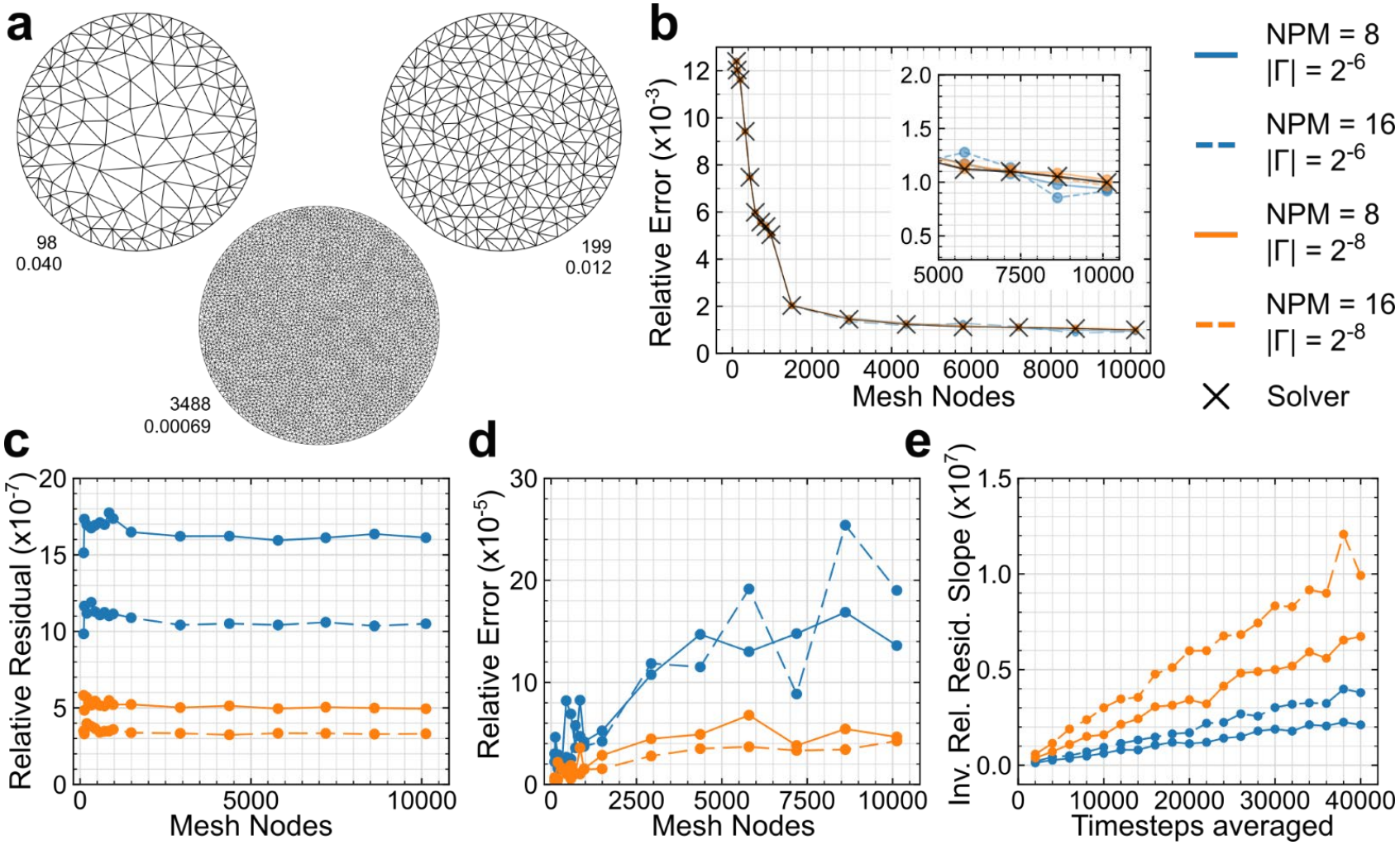
Bradley H. Theilman & James B. Aimone

The finite element method (FEM) is one of the most important and ubiquitous numerical methods for solving partial differential equations (PDEs) on computers for scientific and engineering discovery. Applying the FEM to larger and more detailed scientific models has driven advances in high-performance computing for decades. Here we demonstrate that scalable spiking neuromorphic hardware can directly implement the FEM by constructing a spiking neural network that solves the large, sparse, linear systems of equations at the core of the FEM. We show that for the Poisson equation, a fundamental PDE in science and engineering, our neural circuit achieves meaningful levels of numerical accuracy and close to ideal scaling on modern, inherently parallel and energy-efficient neuromorphic hardware, specifically Intel's Loihi 2 neuromorphic platform. We illustrate extensions to irregular mesh geometries in both two and three dimensions as well as other PDEs such as linear elasticity. Our spiking neural network is constructed from a recurrent network model of the brain's motor cortex and, in contrast to black-box deep artificial neural network based methods for PDEs, directly translates the well-understood and trusted mathematics of the FEM to a natively spiking neuromorphic algorithm.

Theilman and Aimone, Nature Machine Intelligence 2025
https://www.nature.com/articles/s42256-025-01143-2

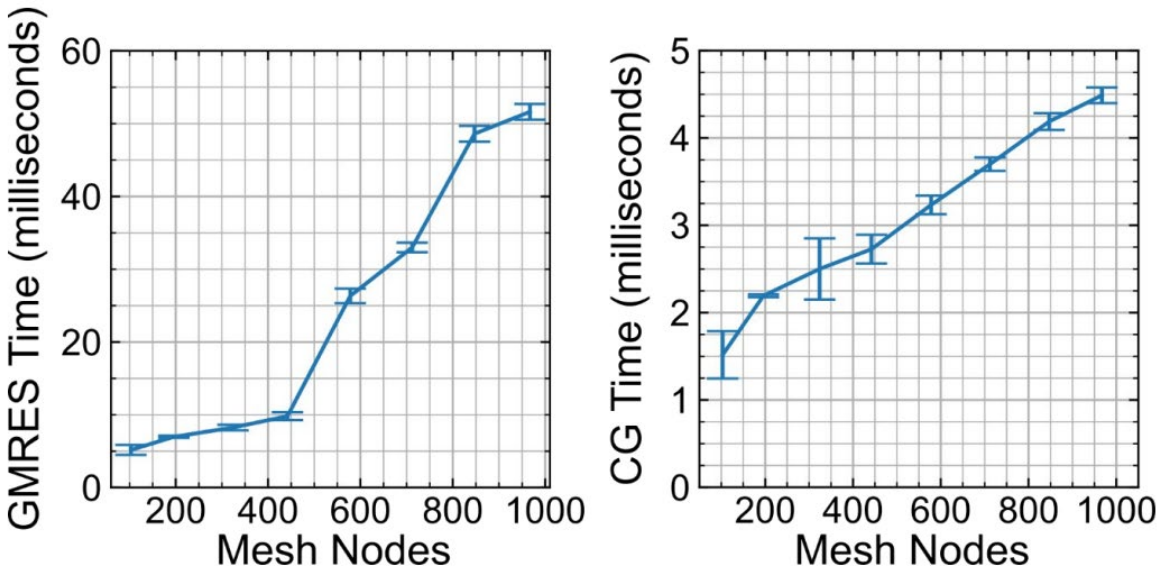
NEUROFEM IN ACTION

NEUROFEM PROVIDES SOLVER-QUALITY SOLUTIONS

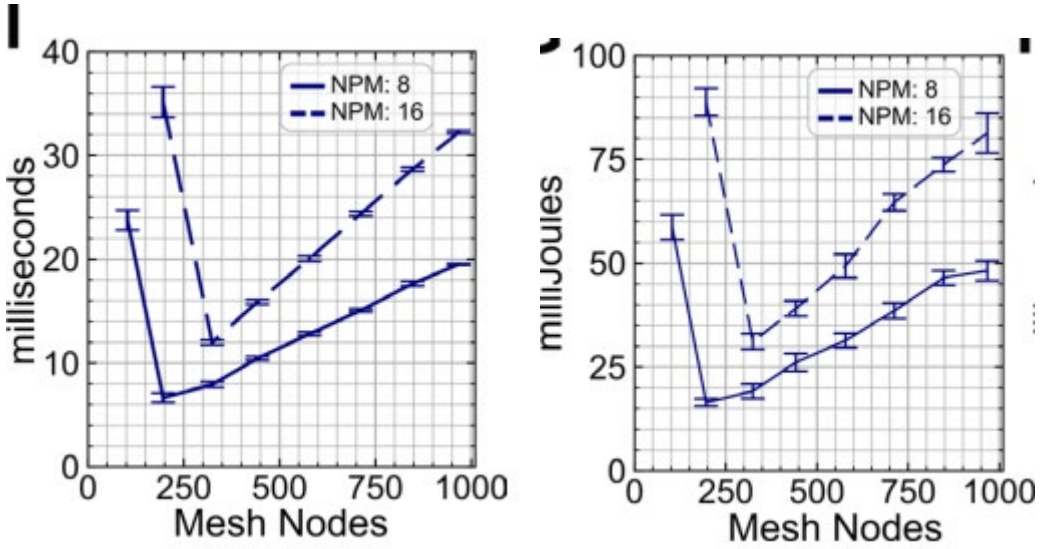


Theilman and Aimone, Nature Machine Intelligence 2025
https://www.nature.com/articles/s42256-025-01143-2

NEUROFEM IS SIMILAR IN SPEED TO GMRES AND CONJUGATE GRADIENT



Neuromorphic solution is general



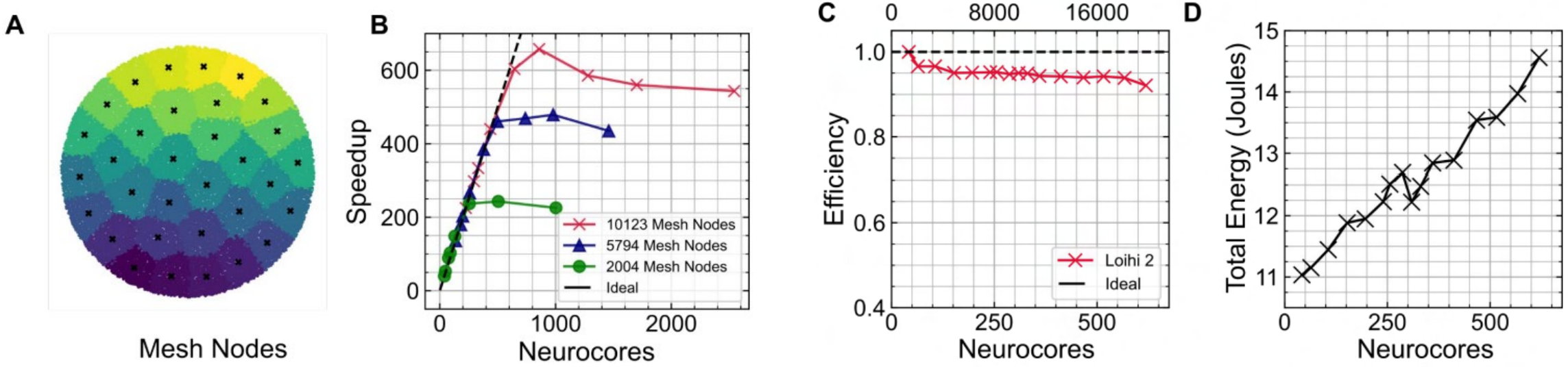
Conjugate gradient is faster at small scales, but can only be used for symmetric matrices

Theilman and Aimone, Nature Machine Intelligence 2025
https://www.nature.com/articles/s42256-025-01143-2

NEUROFEM SHOWS COMPELLING STRONG AND WEAK SCALING

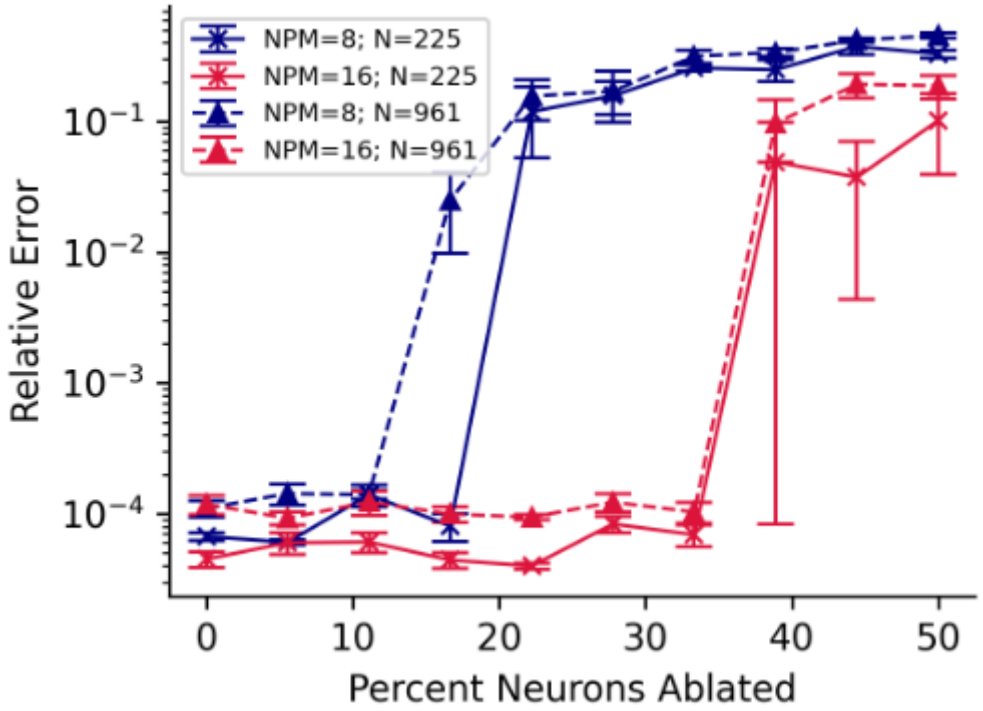
Strong Scaling

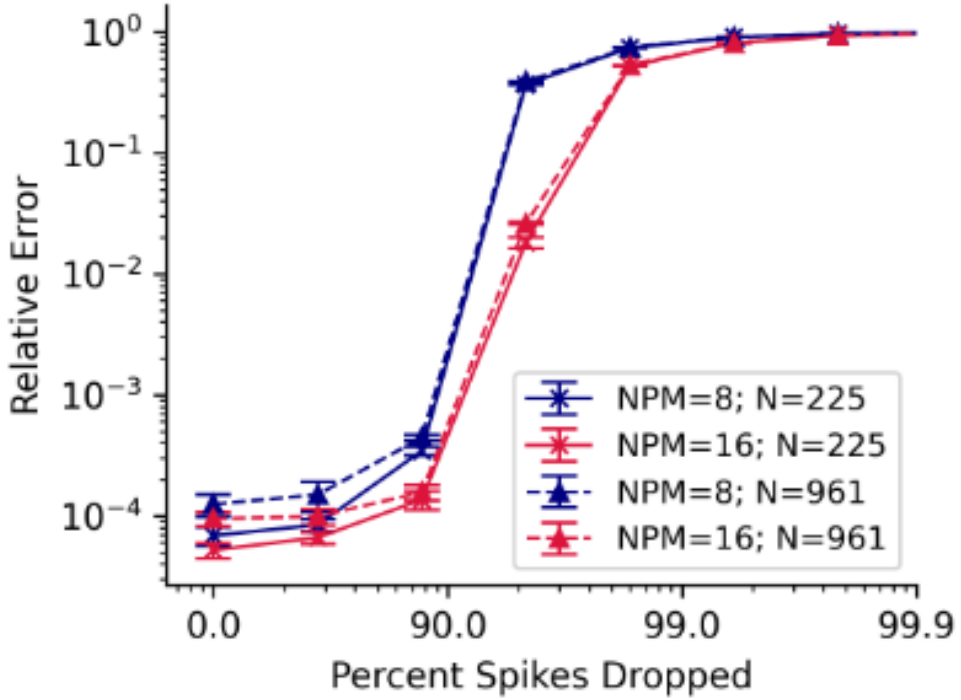
Weak Scaling



Theilman and Aimone, Nature Machine Intelligence 2025
https://www.nature.com/articles/s42256-025-01143-2

NEUROFEM IS INTRINSICALLY FAULT AND ERROR TOLERANT

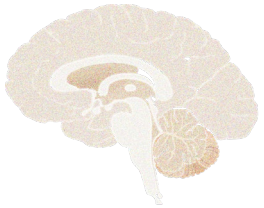


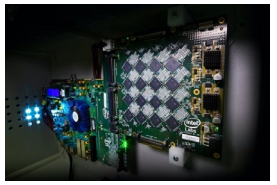


Theilman and Aimone, NICE 2026

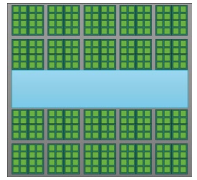
SPARSE LINEAR SOLVERS (NEUROFEM)

The Brain / Neuromorphic





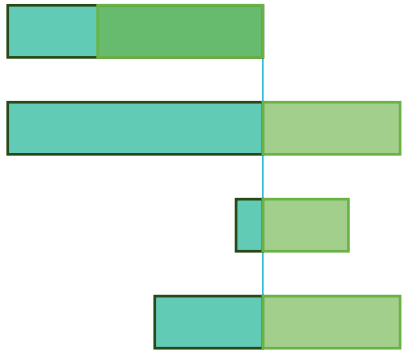
GPUs / SIMD Architectures



Structural Metrics

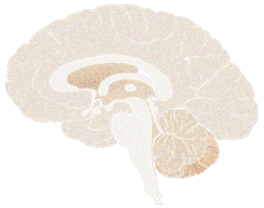
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

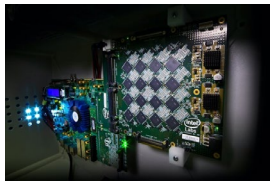
Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

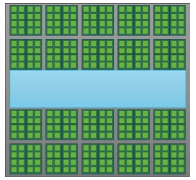
MONTE CARLO

The Brain / Neuromorphic





GPUs / SIMD Architectures



Structural Metrics

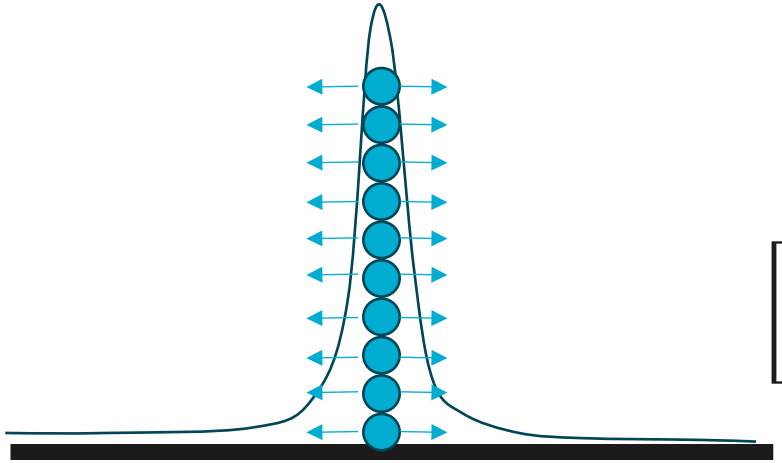
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

CAN WE REFORMULATE MONTE CARLO FOR NEUROMORPHIC?

Initial State

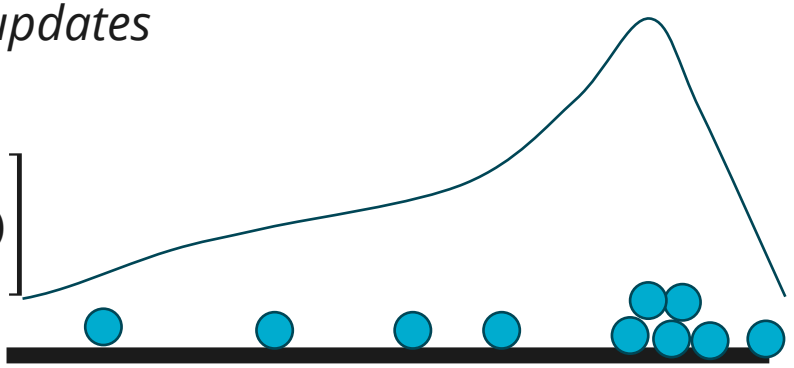


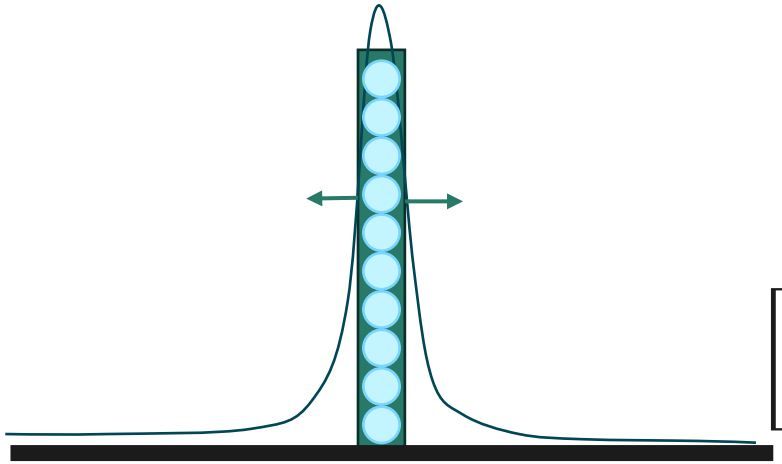
$$dx_i/dt = f(x_i, X, \cdots)$$

K particles *K location-dependent updates*

$$\begin{bmatrix} \uparrow \\ x_i(t+1) \\ \downarrow \end{bmatrix} = f\left(\begin{bmatrix} \uparrow \\ x_i(t) \\ \downarrow \end{bmatrix}, X, \cdots\right) + \begin{bmatrix} \uparrow \\ x_i(t) \\ \downarrow \end{bmatrix}$$

Final State

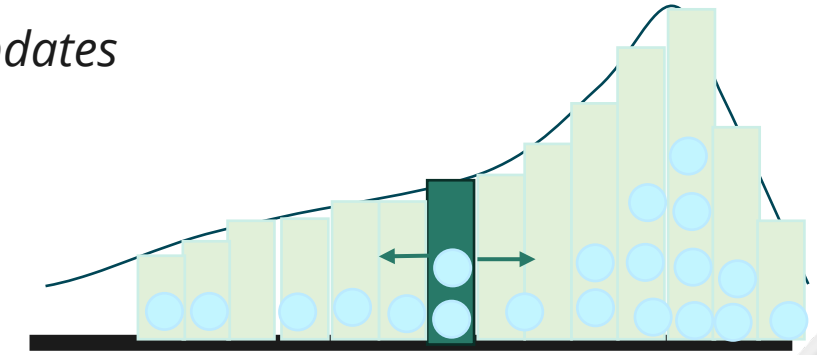




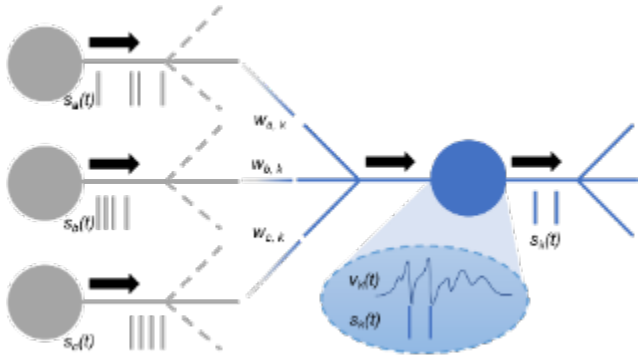
$$dm_i/dt = g_i(X, \cdots)$$

M locations *M location-specific updates*

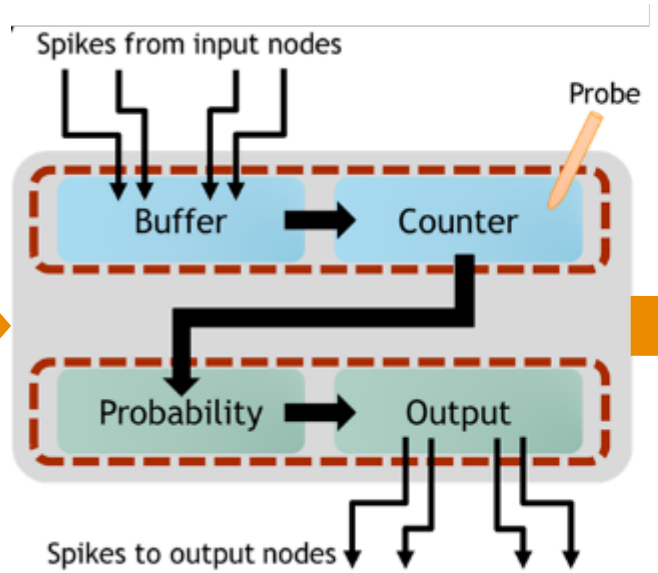
$$\begin{bmatrix} \uparrow \\ m_i(t+1) \\ \downarrow \end{bmatrix} = g_i(X, \cdots) + \begin{bmatrix} \uparrow \\ m_i(t) \\ \downarrow \end{bmatrix}$$

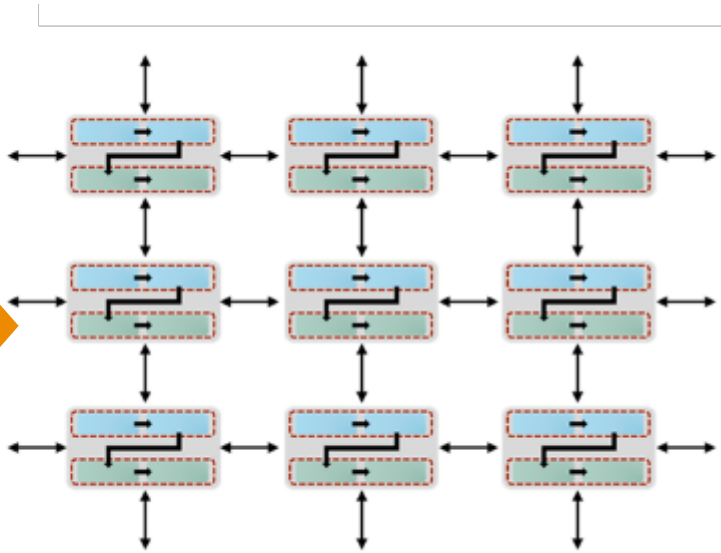


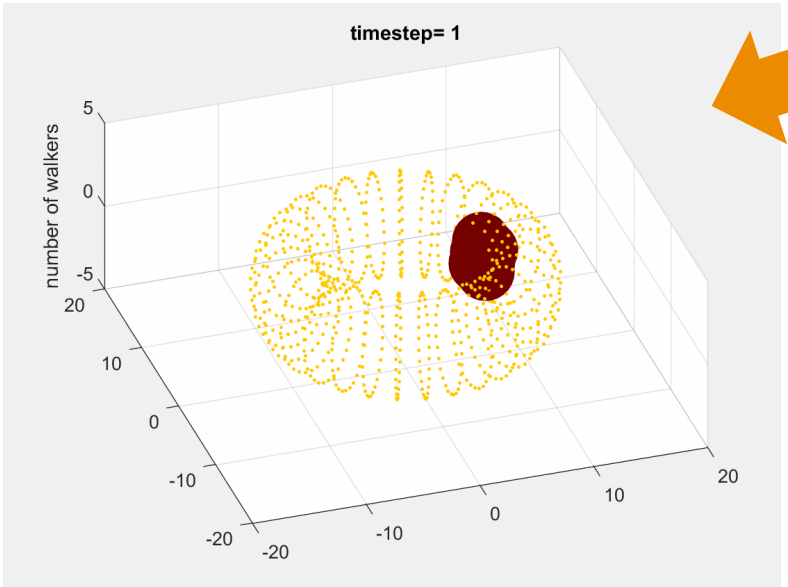
USE NEURONS TO REPRESENT *STATE SPACE* OF MONTE CARLO AND USE SPIKES TO REPRESENT PARTICLES

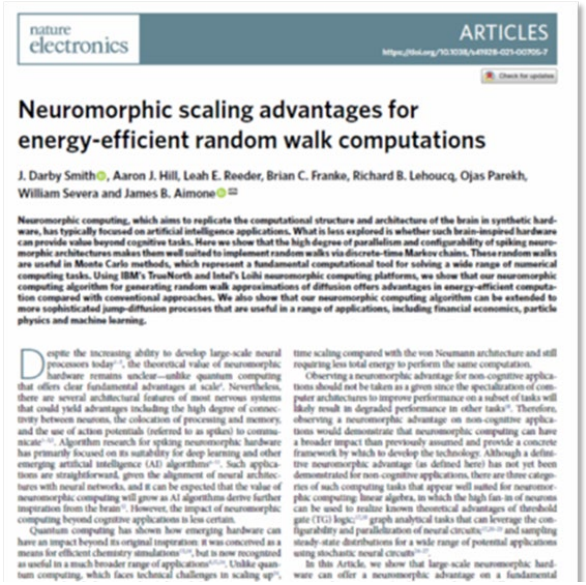


Leaky integrate-and-fire neuron









nature electronics

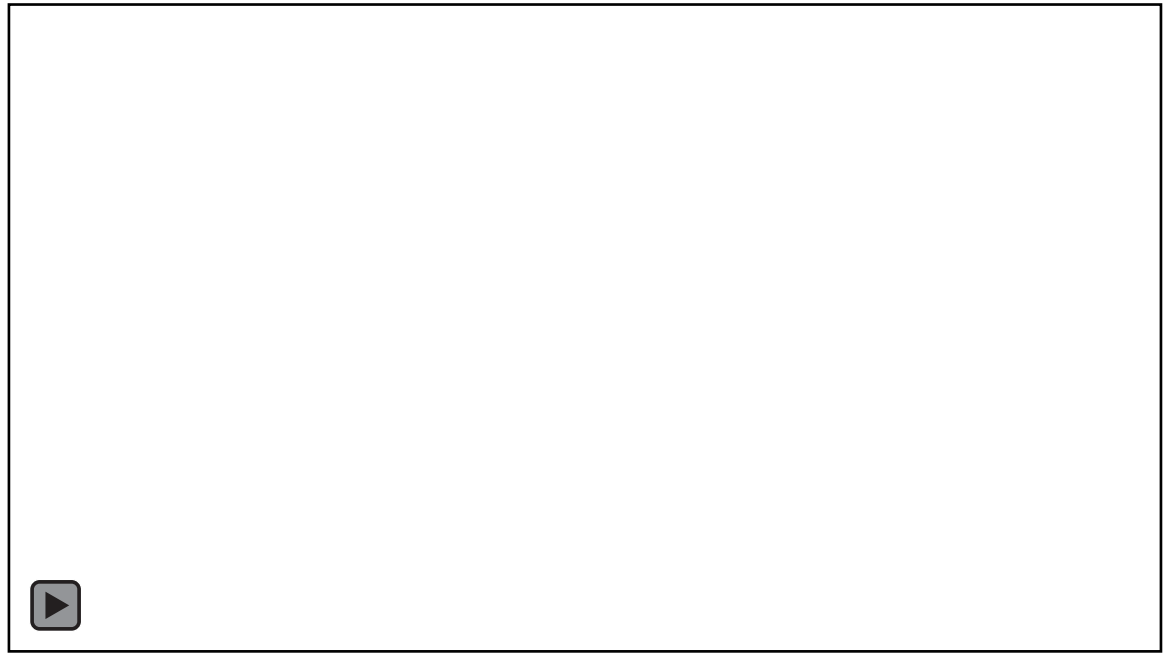
ARTICLES

Neuromorphic scaling advantages for energy-efficient random walk computations

J. Darby Smith, Aaron J. Hill, Leah E. Reeder, Brian C. Franke, Richard B. Lehoucq, Ojas Parekh, William Severa and James B. Aimone

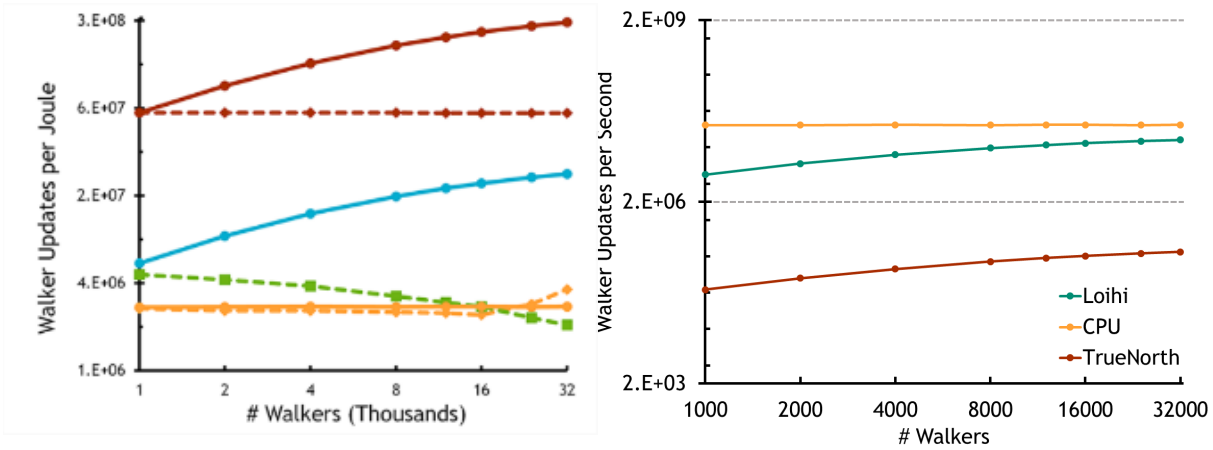
Smith et al., Nature Electronics 2022

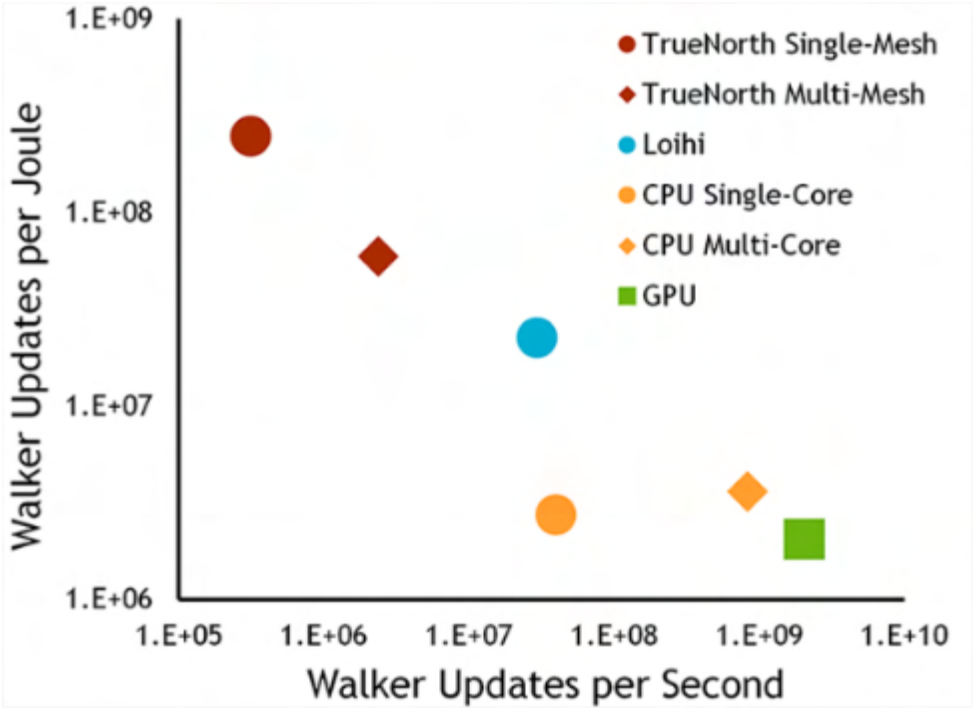
NEUROMORPHIC COMPUTING ADVANTAGE APPEARS TO BE WHEN AN ALGORITHM CAN SPLIT THE TASK ACROSS COMPUTATIONAL GRAPH WITH SPARSE COMMUNICATION



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

WHAT PDES CAN NEURAL RANDOM WALKS ADDRESS?

Class of Partial Integro-Differential Equations:

$$\frac{\partial}{\partial t}u(t,\boldsymbol{x}) = \frac{1}{2}\sum_{i,j}(\boldsymbol{a}\boldsymbol{a}^{\top})_{i,j}(t,\boldsymbol{x})\frac{\partial^2}{\partial x_i \partial x_j}u(t,\boldsymbol{x}) + \sum_i b_i(t,\boldsymbol{x})\frac{\partial}{\partial x_i}u(t,\boldsymbol{x})$$
$$+\lambda(t,\boldsymbol{x})\int\Big(u\big(t,\boldsymbol{x}+\boldsymbol{h}(t,\boldsymbol{x},q)\big)-u(t,\boldsymbol{x})\Big)\phi_Q(q;t,\boldsymbol{x})\mathrm{d}q$$
$$+c(t,\boldsymbol{x})u(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), \qquad x\in\mathbb{R}^d, t\in[0,\infty).$$

Stochastic Process:

NMC Hardware Simulates This Stochastic Process

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

Monte Carlo Approximates This Expectation

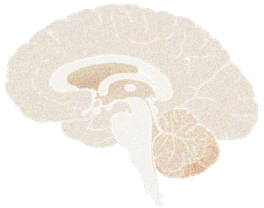
Solution to initial value problem (u(0,x)=g(x)):

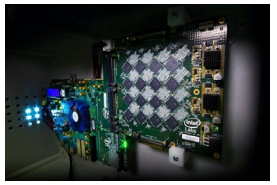
$$u(t,\boldsymbol{x}) = \mathbb{E}\left[g\big(\boldsymbol{X}(t)\big)\exp\left(\int_0^t c\big(s,\boldsymbol{X}(s)\big)\mathrm{d}s\right) + \int_0^t f\big(s,\boldsymbol{X}(s)\big)\exp\left(\int_0^s c\big(\ell,\boldsymbol{X}(\ell)\big)\mathrm{d}\ell\right)\mathrm{d}s \,\middle|\, \boldsymbol{X}(0)=\boldsymbol{x}\right].$$

Smith et al., Nature Electronics 2022

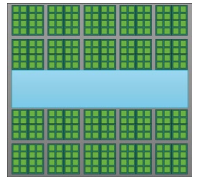
MONTE CARLO

The Brain / Neuromorphic





GPUs / SIMD Architectures



Structural Metrics

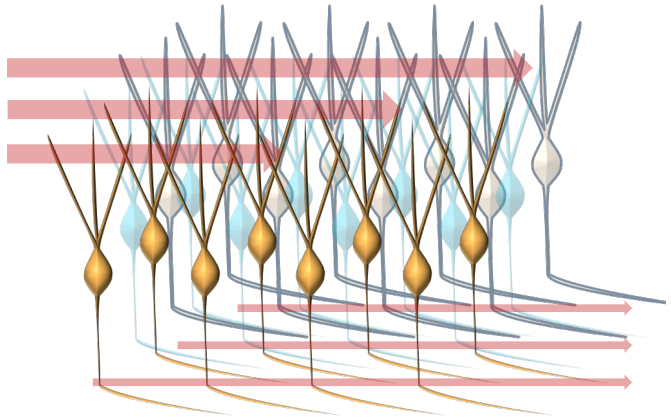
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

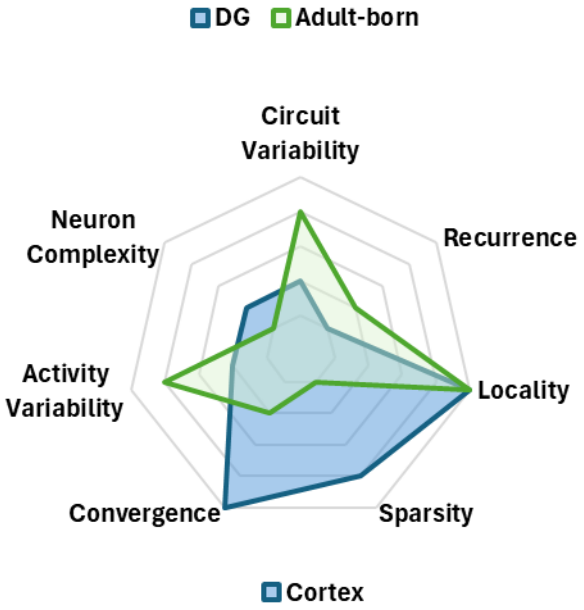
Trace Metrics

Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

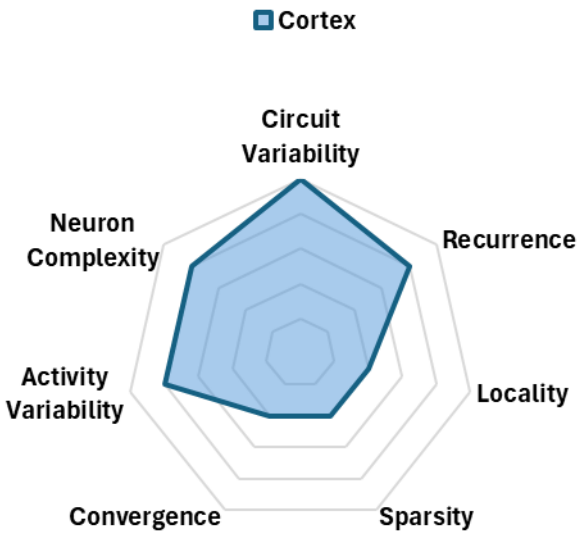
WHAT ABOUT THE BRAIN?

Dentate Gyrus

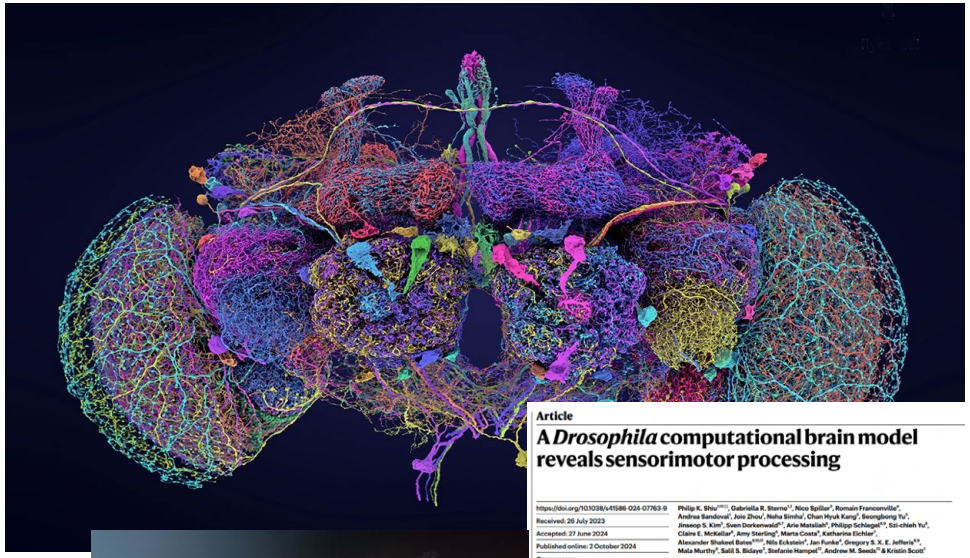




Cortex



REAL BRAINS HAVE NON-TRIVIAL STRUCTURE



Article

A *Drosophila* computational brain model reveals sensorimotor processing

https://doi.org/10.1038/s41586-024-07763-9
Received: 26 July 2023
Accepted: 27 June 2024
Published online: 2 October 2024
Open access
Check for updates

Philip K. Shiu[1,2,3], Gabriella R. Sterne[1,2], Nico Spiller[1], Romain Franconville[4], Andrea Sandoval[1], Joie Zhou[1], Neha Simha[1], Chan Hyuk Kang[5], Seongbong Yu[5], Jinseop S. Kim[6], Sven Dorkenwald[7,8], Arie Matsliah[9], Philipp Schlegel[6,10], Sai-chieh Yu[1], Claire E. McKellar[11], Amy Sterling[12], Marta Costa[6], Katharina Eichler[6], Alexander Shakeel Bates[6,13,14], Nils Eckstein[6], Jan Funke[6], Gregory S. X. E. Jefferis[6,15], Mala Murthy[16], Salil S. Bidaye[3], Stefanie Hampel[17], Andrew M. Seeds[18] & Kristin Scott[1]

The recent assembly of the adult *Drosophila melanogaster* central brain connectome, containing more than 125,000 neurons and 50 million synaptic connections, provides a template for examining sensory processing throughout the brain[1–3]. Here we create a leaky integrate-and-fire computational model of the entire *Drosophila* brain, on the basis of neural connectivity and neurotransmitter identity[3], to study circuit properties of feeding and grooming behaviours. We show that activation of sugar-sensing or water-sensing gustatory neurons in the computational model accurately predicts neurons that respond to tastes and are required for feeding initiation[4]. In addition, using the model to activate neurons in the feeding region of the *Drosophila* brain predicts those that elicit motor neuron firing[5]—a testable hypothesis that we validate by optogenetic activation and behavioural studies. Activating different classes of gustatory neurons in the model makes accurate predictions of how several taste modalities interact, providing circuit-level insight into aversive and appetitive taste processing. Additionally, we applied this model to mechanosensory circuits and found that computational activation of mechanosensory neurons predicts activation of a small set of neurons comprising the antennal grooming circuit, and accurately describes the circuit response upon activation of different mechanosensory subtypes[6–10]. Our results demonstrate that modelling brain circuits using only synapse-level connectivity and predicted neurotransmitter identity generates experimentally testable hypotheses and can describe complete sensorimotor transformations.

The *Drosophila melanogaster* central brain connectome, comprising more than 125,000 neurons and 50 million synaptic connections, allows brainwide analyses of how the fly processes sensory information[1,2,11–16]. Although the *Drosophila* brain is comparatively small, a single *Drosophila* neuron may be connected synaptically to hundreds of downstream neurons, and interpreting how this connectivity relates to behaviour is not straightforward.

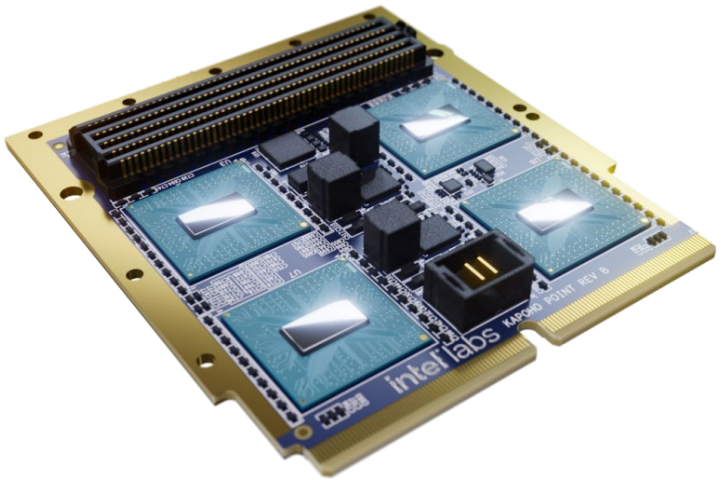
To model the neural circuit mechanisms that generate behaviour, we implement a simple leaky integrate-and-fire model using the connection weights derived from the entire adult *Drosophila* central brain connectome of reconstructed electron microscopy neurons[1,2,17–19], as well as neurotransmitter predictions for each neuron[3,20]. In this model, spiking of a neuron alters the membrane potential of downstream neurons in proportion to the connectivity from the upstream neuron[21] (Fig. 1a and Methods): if a downstream neuron's membrane potential reaches the firing threshold, that neuron, in turn, fires.

We implemented this model in the spiking neural network simulator Brian2 (ref. 22) using all Flywire neurons. The baseline firing of each neuron in our model is 0 Hz. All biophysical parameters are taken from previous *D. melanogaster* modelling or electrophysiology efforts[6,10], from the synaptic weights from the Flywire connectome[3,20] and from the neurotransmitter predictions[3], except for W_{syn}—the single free parameter of the model, which corresponds to the magnitude of the change in downstream membrane potential that results from a single excitatory or inhibitory synapse (Methods). By driving activity in a sparse set of neurons, the model predicts changes in downstream firing.

We examined the model's ability to predict circuit activity in two systems: feeding initiation and antennal grooming. We began by examining

[1]Department of Molecular and Cell Biology and Helen Wills Neuroscience Institute, University of California, Berkeley, CA, USA. [2]University of Rochester Medical Center, Department of Biomedical Genetics, New York, NY, USA. [3]Max Planck Florida Institute for Neuroscience, Jupiter, FL, USA. [4]HHMI Janelia Research Campus, Ashburn, VA, USA. [5]Department of Biological Sciences, Sungkyunkwan University, Suwon, South Korea. [6]Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA. [7]Computer Science Department, Princeton University, Princeton, NJ, USA. [8]Department of Zoology, University of Cambridge, Cambridge, UK. [9]Neurobiology Division, MRC Laboratory of Molecular Biology, Cambridge, UK. [10]Centre for Neural Circuits and Behaviour, The University of Oxford, Oxford, UK. [11]Department of Neurobiology and Howard Hughes Medical Institute, Harvard Medical School, Boston, MA, USA. [12]Institute of Neurobiology, University of Puerto Rico Medical Sciences Campus, San Juan, Puerto Rico. [13]Present address: Econ Systems, San Francisco, CA, USA. [14]e-mail: philshiu@gmail.com

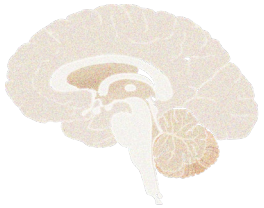
125,000 neurons
50 million synapses

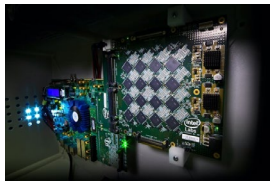


Shiu et al., Nature 2024

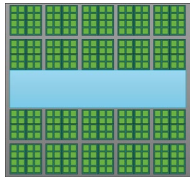
BRAIN SIMULATIONS ARE THE IDEAL APPLICATION

The Brain / Neuromorphic





GPUs / SIMD Architectures



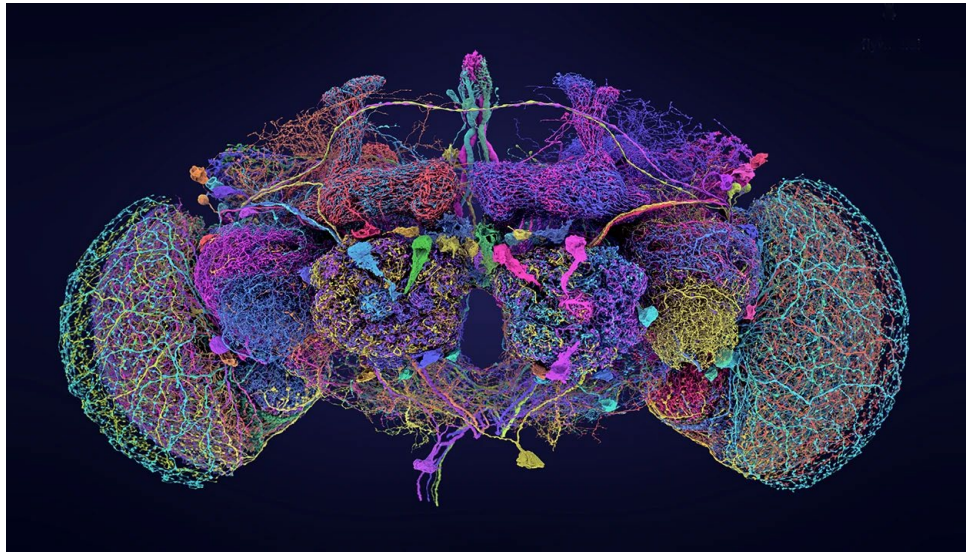
Structural Metrics

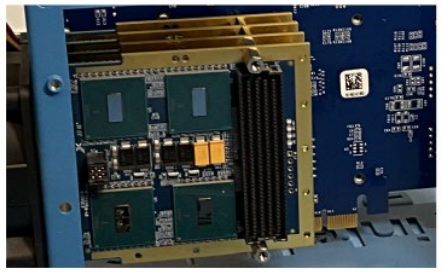
Structural Heterogeneity		Structural Homogeneity
Recurrence / Structural Reuse		Feed-forward / Acyclic
Local Connectivity		Global Connectivity

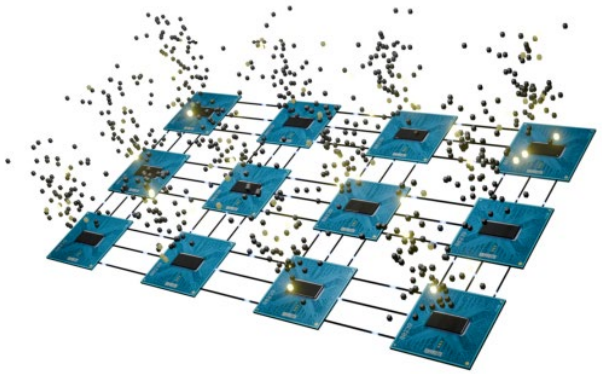
Trace Metrics

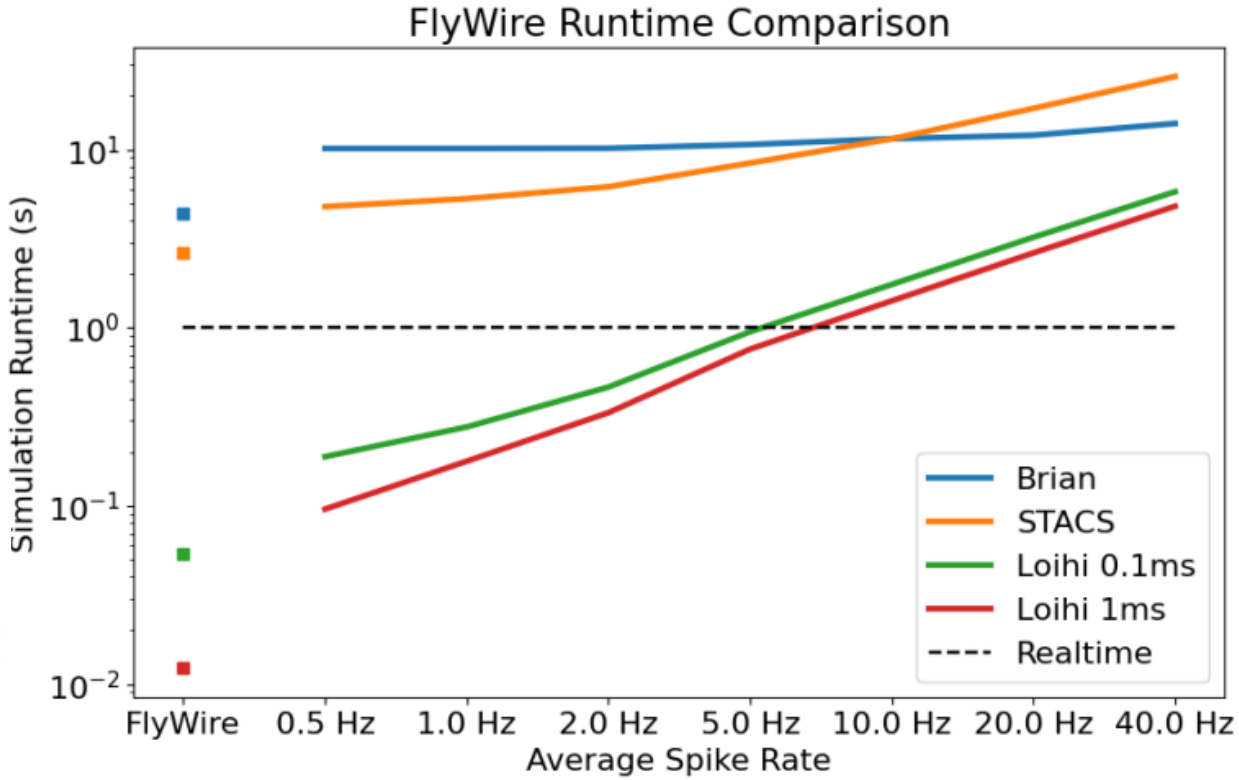
Sparse Activity		Dense Activity
Time-dependent Activity		Time-homogeneous Activity
Thread Divergence		Thread Convergence
Stochasticity okay		Stochasticity challenging

ALGORITHM MATCHES THE ARCHITECTURE: NEUROMORPHIC HARDWARE CAN ACCELERATE BIOLOGICAL SIMULATIONS BY >100X







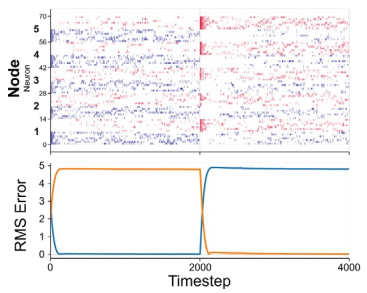


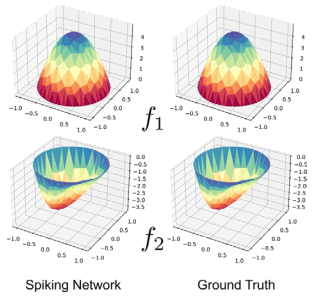
Wang et al., Arxiv 2025
https://arxiv.org/pdf/2508.16792

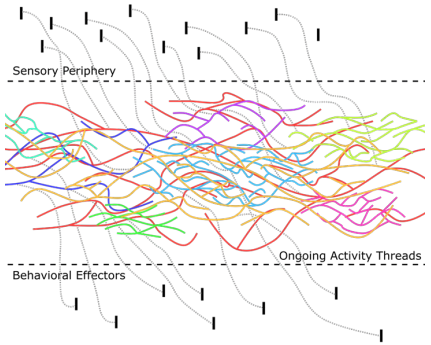
The **algorithm** must match the **architecture**.

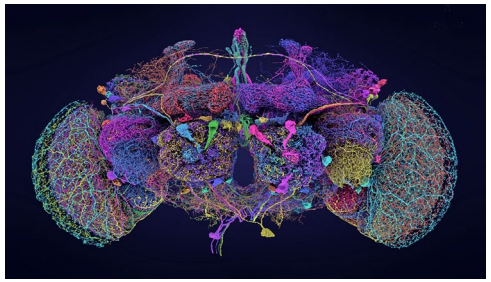
- We must break out of representation-centric connectionist mindset
- Algorithms based on trajectories of spikes over time and space?
- Is the goal to understand uncertainties and distributions?
- *Time to revisit what learning means?*

- Processing in memory
- Event-driven
- Local & sparse activity
- *Heterogeneity, stochasticity, divergence okay!*









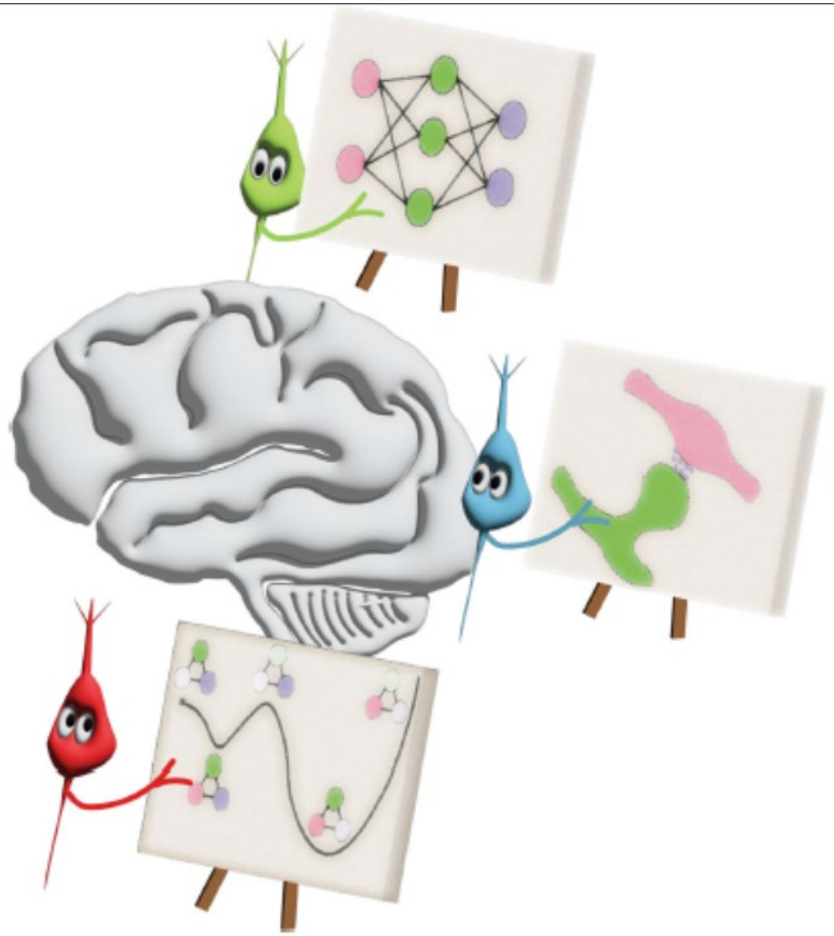
THANK YOU!

Any questions to jbaimon@sandia.gov

Neural Exploration & Research Lab
COGNITIVE & EMERGING COMPUTING

Craig M Vineyard, Suma G Cardwell, Corinne M Teeter, William Severa, J Darby Smith, Felix Wang, Fred Rothganger, Michael Krygier, Cale Crowder, Bradley H Theilman, William Chapman, Ryan Dellana, Mark Plagge, Efrain Gonzalez, Srideep Musuvathy

Aaron Hill, Shashank Misra, Yang Ho, Brady Taylor, Chris Allemang, Rich Lehoucq, Brian Franke, Ojas Parekh



U.S. DEPARTMENT OF ENERGY | Office of Science

LDRD LABORATORY DIRECTED RESEARCH & DEVELOPMENT

Sandia National Laboratories

ASC™