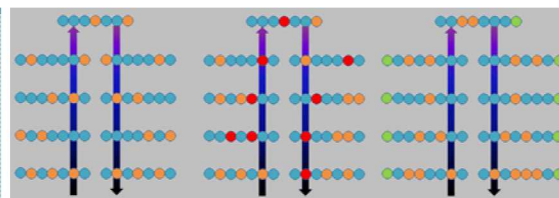
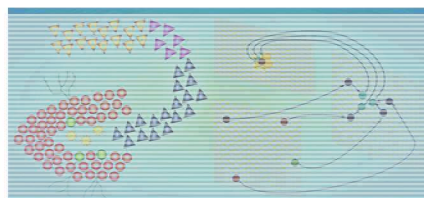
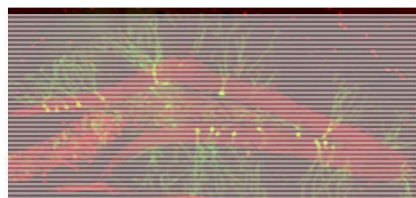


# Computing with Spikes

*Everything from the deep learning to numerical applications*



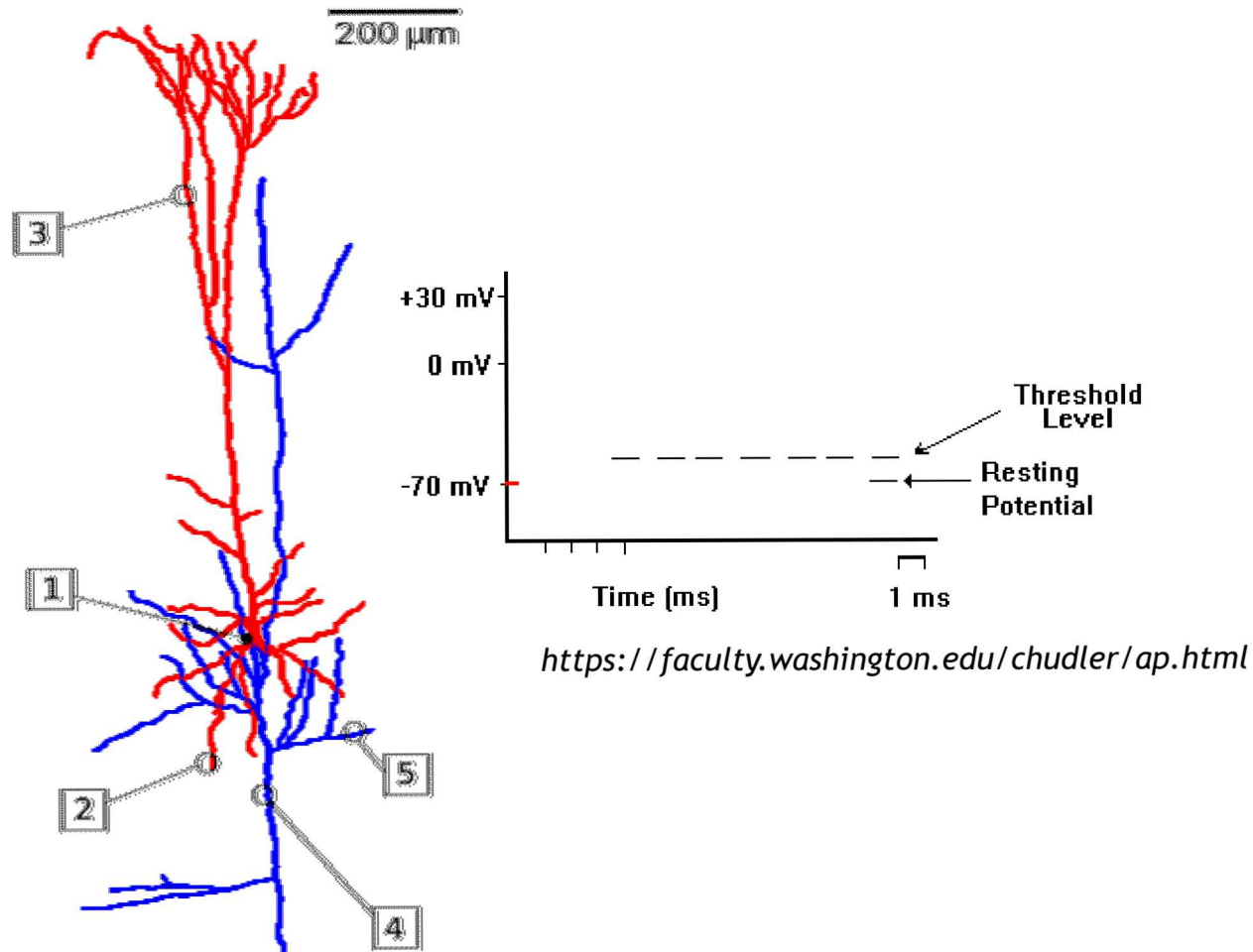
PRESENTED BY

Brad Aimone

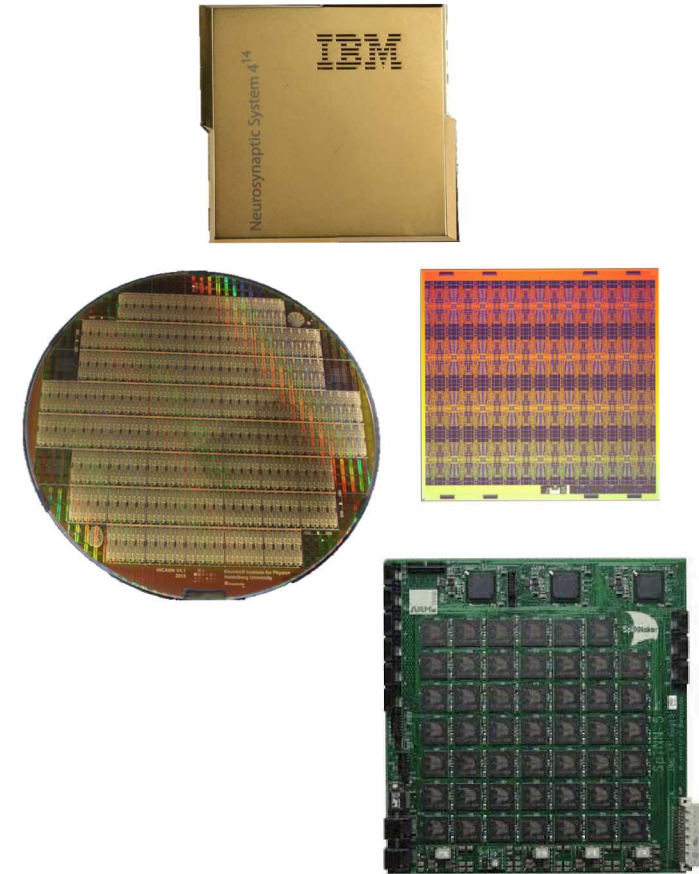


Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# What is spiking?



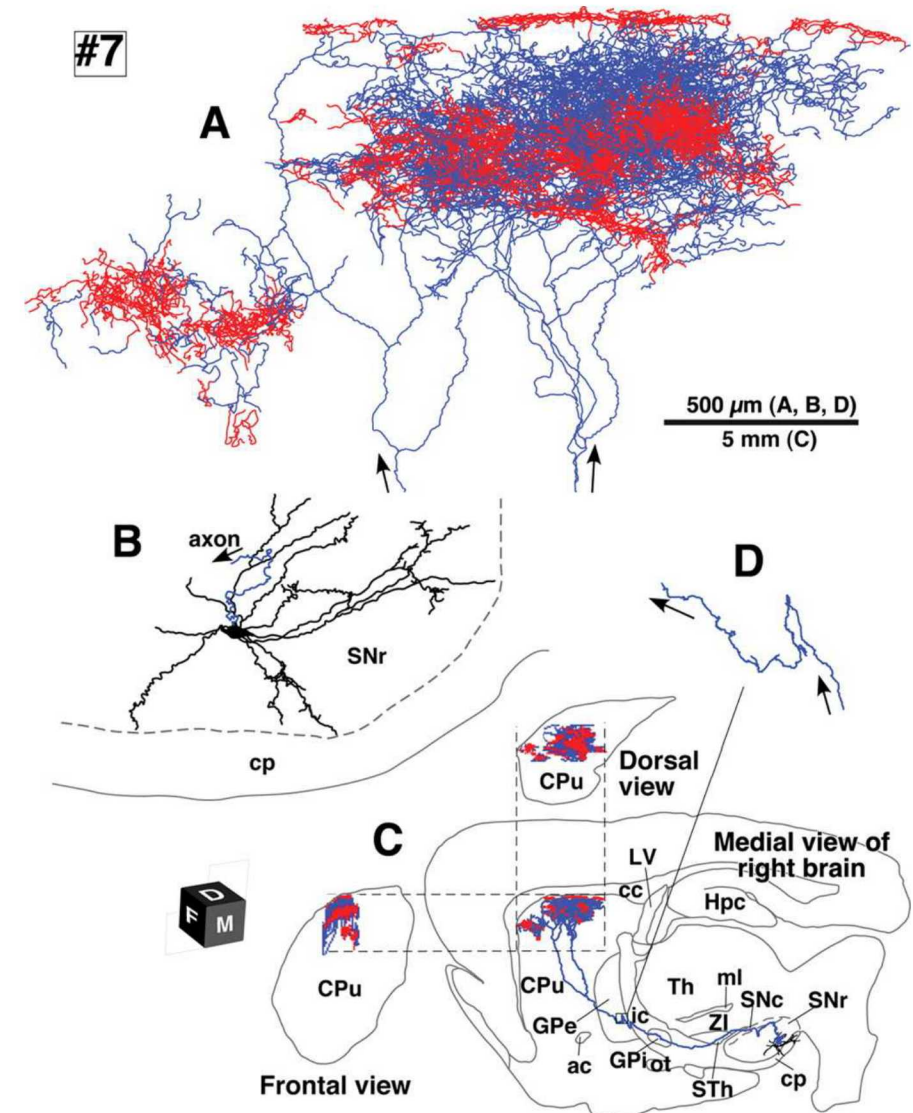
Pyramidal Cell -- Wikipedia



Clockwise from top: IBM TrueNorth, Intel Loihi, SpiNNaker, BrainScales

# Why spiking?

- Event-driven
  - Only expend energy when neuron crosses threshold
- Reliable and efficient over long distances
  - Neurons often project across brain or whole body...
- Robust to noise
  - Away from threshold, biophysical noise should not accidentally cause spikes



# Correcting some common misconceptions about spiking

- *Spiking is necessary for brain-like computation*

- Reality: The advantage of spiking is efficiency and reliability over distance, not computability.

It changes the tradeoffs between time, power, and space

- *Spiking does not offer anything for algorithms*

- Reality: Spiking facilitates developing algorithms that more directly leverage time in computing

- *Spiking reduces the accuracy of algorithms*

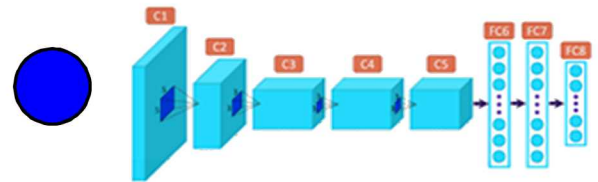
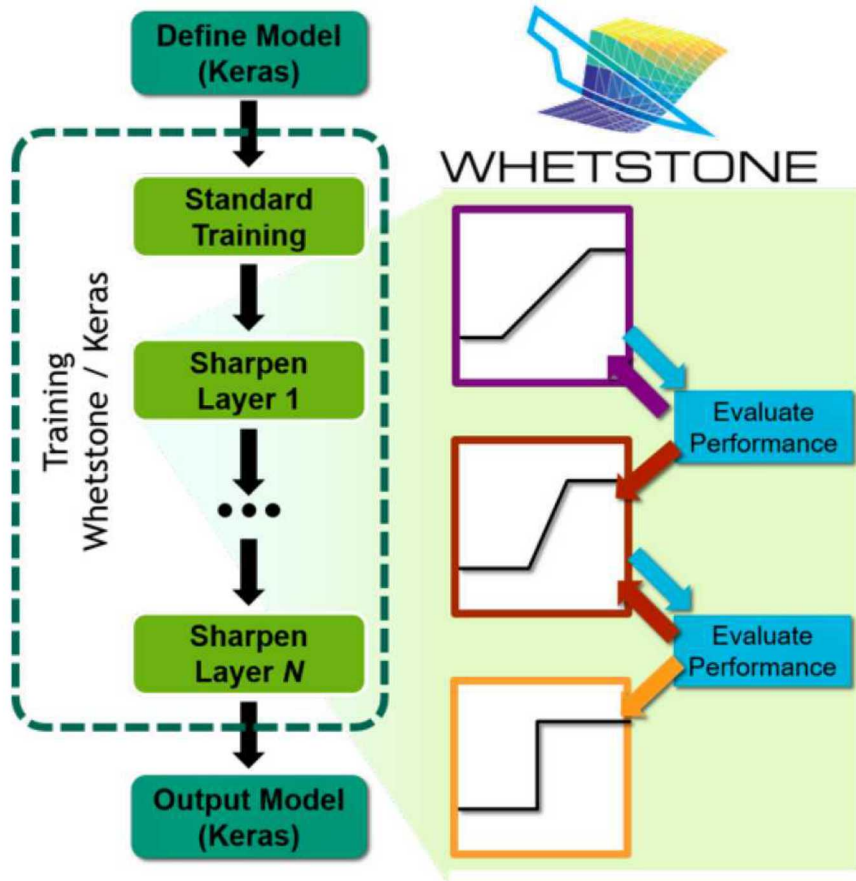
- Reality: Not necessarily! Spiking does lower the precision of communication, but often this precision is unnecessary or can be compensated for in other ways.

- *Spiking requires paying a time penalty*

- Reality: Not always! Some coding schemes are actually time advantageous – e.g., you can implement very fast threshold gate circuit algorithms on spiking hardware

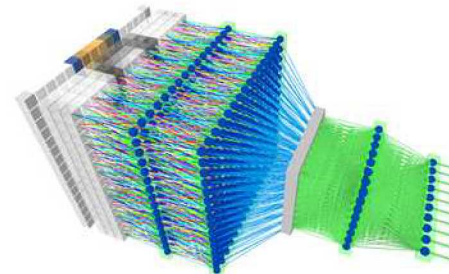


# What can you do with spiking neurons?



## Spiking deep neural networks

- Whetstone allows us to use spiking communication with *no time penalty* and minimal accuracy reduction



ARTICLES  
nature machine intelligence

### Training deep neural networks for binary communication with the Whetstone method

William Severa<sup>1</sup>, Craig M. Vineyard<sup>2</sup>, Ryan DeLana<sup>1</sup>, Stephen J. Verzi<sup>1</sup> and James B. Aumane<sup>1</sup>\*

The computational cost of deep neural networks presents challenges to broadly deploying these algorithms. Low-power and energy-efficient neuromorphic processors offer potentially dramatic performance per watt improvements over traditional processors. However, programming these brain-inspired platforms generally requires platform-specific expertise. It is therefore difficult to achieve the degree of performance on these platforms, limiting their applicability in the present. Whetstone is a method to bridge this gap by converting deep neural networks to have discrete, binary communication. During the training process, the activation function of each layer is progressively sharpened towards a threshold activation, with limited loss in performance. Whetstone sharpened networks do not require a rate code or other spike-based coding scheme, thus producing networks compatible in timing and data to conventional artificial neural networks. We demonstrate Whetstone on a number of architectures and tasks such as image classification, autonomous and semantic segmentation. Whetstone is currently implemented within the Keras framework for TensorFlow and is widely extensible.

Artificial neural networks (ANNs) algorithms, specifically deep convolutional networks (DCNs) and other deep learning methods, have become the state-of-the-art techniques for a number of machine learning applications<sup>1–4</sup>. While deep learning models can be expensive both in time and energy to operate and even more expensive to train, their exceptional accuracy on hands-on machine learning tasks such as image classification and object processing has made their use essential in many domains.

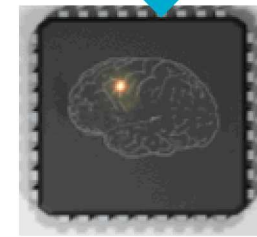
Many applications can rely on generic servers to perform deep learning calculations. However, for many applications such as onboard processing in autonomous platforms that self-driving cars, drones and smart phones, the resource requirements of running large ANNs may still prove to be prohibitive<sup>5</sup>. Large ANNs with many parameters require a significant storage capacity that is not always available, and data movement energy costs are greater than that of performing the computation, making large ANNs impractical for edge processing. Additionally, onboard processing capabilities are often limited by energy budget requirements for the computing hardware in the target device. Other factors such as privacy and data sharing also provide motivation for performing computation locally rather than on a remote server.

The development of specialized hardware to enable more efficient ANN calculations seeks to facilitate running ANNs on resource-constrained environments, particularly for neural algorithms that simply require the deployment of an inference-ready network. A common approach today is to optimize for compact neural networks of ANNs in application-specific integrated circuits (ASICs)<sup>6–8</sup>. However, while these ANNs can provide substantial acceleration, their power costs are still high for some embedded applications and often lack flexibility for implementing alternative ANN architectures.

Brain-inspired neuromorphic hardware presents an alternative to conventional ASIC accelerators, and has been shown to be capable of running ANNs with potentially orders-of-magnitude lower power consumption (that is, performance per watt). The hardware of neuromorphic hardware is typically evolving<sup>9–11</sup>, however, increasingly these approaches leverage existing silicon semiconductor energy savings. Neuromorphic spiking, which emulates all or some active potentials in biological neurons, limits communication in hardware only to discrete events. For spiking neuromorphic hardware to be useful, however, it is necessary to convert an ANN, for which time-multiplexed binary communication is the native language, to a spiking neural network (SNN) implementation. This provides for the ability of spiking and ANN hardware.

The conversion of ANNs to SNNs—whenever their form—is an arduous task, as ANNs depend on gradient-based backpropagation training algorithms, which require high-precision communications, and the resulting networks effectively ignore the presence of their precision. While there are methods for converting existing ANNs to SNNs, these transformations often require using approximations that diminish the benefits of spiking. Here, we describe a new approach to training SNNs, where the ANN training is in no way altered, but the output is converted to a SNN in the process. Specifically, the training procedure can include the removal of some or all of the precision of the ANN, while the output remains a SNN. This method, which we term Whetstone (Fig. 1), inspired by the tool to sharpen a dull knife, is intuitively agnostic to both the type of ANN being trained and the target neuromorphic hardware. Indeed, the intent is to provide a straightforward interface for machine learning researchers to leverage the potential capabilities of low-power neuromorphic hardware on a wide range of deep learning applications (see section ‘Implementation and software package details’).

**Results** Whetstone method converts general ANNs to SNNs. The Whetstone algorithm operates by incorporating the conversion into binary activation directly into the training process. Because most techniques to train ANNs rely on stochastic gradient descent methods, it is necessary that the activation is non-linear and differentiable during the training process. However, as networks become trained, the training process is able to incorporate non-differentiable constraints, such as a step function, into the communication between nodes. With this shift of the optimization target to

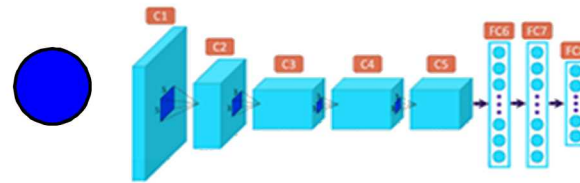
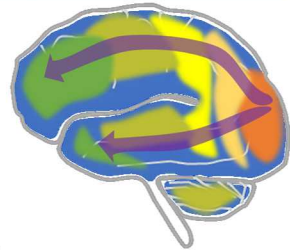
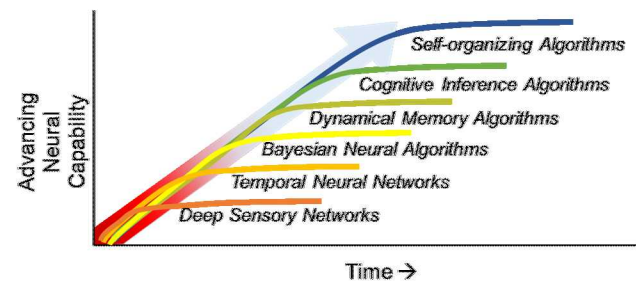


Severa et al., Nature Machine Intelligence, Feb 2019  
Vineyard et al., NICE Proceedings, 2019

# What can you do with spiking neurons?



Algorithm Class	Current Algorithms	Inspiration	Application
Deep Vision Processing	Deep Convolutional Networks (VGG, AlexNet, GoogleNet, etc.), HMax, Neocognitron	Hierarchy of sensory nuclei and early sensory cortices	Static feature extraction (e.g., images) & pattern classification
Temporal Neural Networks	Deep Recurrent Networks (e.g., long short-term memory), Hopfield Networks	Local recurrence of most biological neural circuits, especially higher sensory cortices	Dynamic feature extraction (e.g., videos, audio) & classification
Bayesian Neural Algorithms	Predictive Coding, Hierarchical Temporal Memory, Recursive Cortical Networks	Substantial reciprocal feedback between "higher" and "lower" sensory cortices	Inference across spatial and temporal scales
Dynamical Memory and Control Algorithms	Liquid State Machines, Echo State Networks, Neural Engineering Framework	Continual dynamics of hippocampus, cerebellum, and prefrontal and motor cortices	Online learning content-addressable memory & adaptive motor control
Cognitive Inference Algorithms	Reinforcement learning (e.g., Deep Q-learning), Neural Turing Machines	Integration of multiple modalities and memory into prefrontal cortex, which provides top-down influence on sensory processing	Context and experience dependent information processing and decision making
Self-organizing Algorithms	Neurogenesis Deep Learning	Initial development and continuous refinement of neural circuits to specific input and outputs	Automated neural algorithm development for unknown input and output transformations

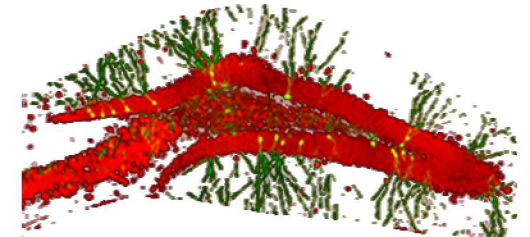


## Spiking deep neural networks

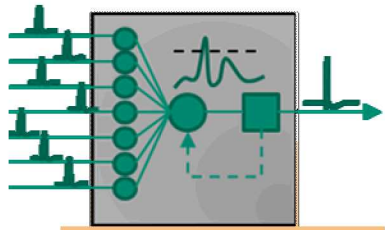
- Whetstone allows us to use spiking communication with *no time penalty* and minimal accuracy reduction

## Neuroscience-constrained algorithms

- Computation incorporates broad range of neural plasticity and dynamics
- *Generally still unexplored from algorithms perspective*

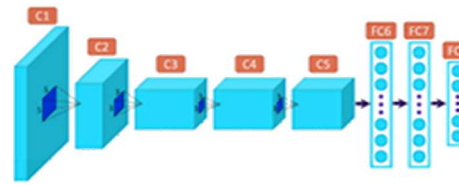


# What can you do with spiking neurons?



*Treat neurons as powerful logic gates*

*Algorithms are circuits...*

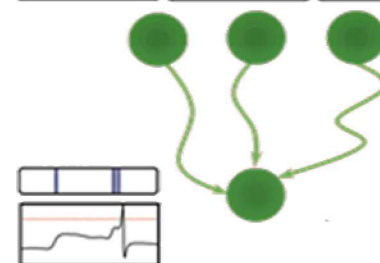
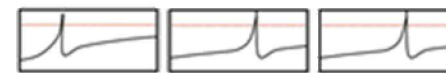
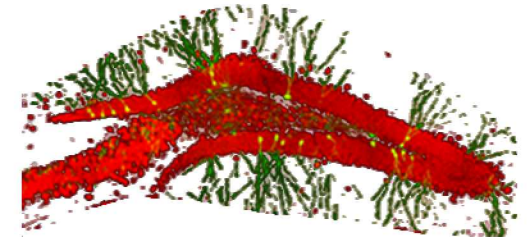


## Spiking deep neural networks

- Whetstone allows us to use spiking communication with *no time penalty* and minimal accuracy reduction

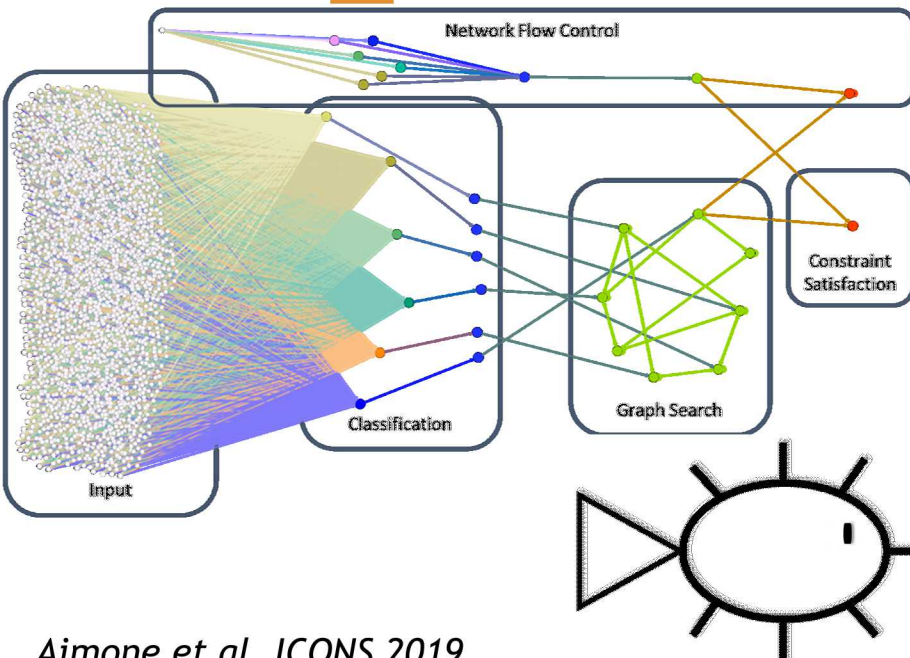
## Neuroscience-constrained algorithms

- Computation incorporates broad range of neural plasticity and dynamics
- *Generally still unexplored from algorithms perspective*



## Spiking neural algorithms

- Hand-crafted circuits of spiking neurons
- Model of parallel computation
- Energy efficiency through event-driven communication and high fan-in logic



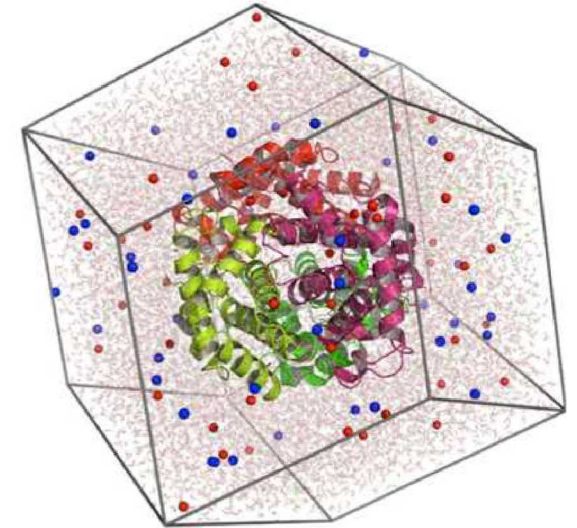
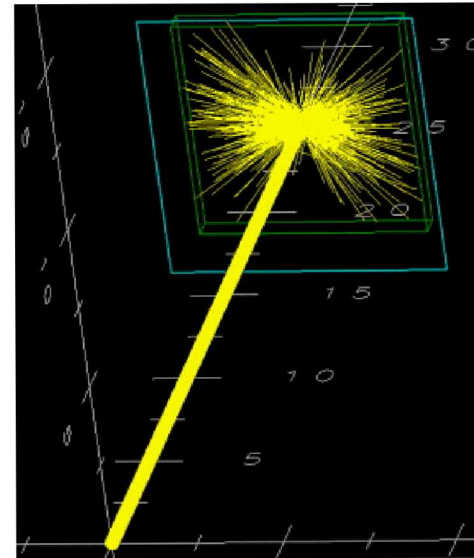
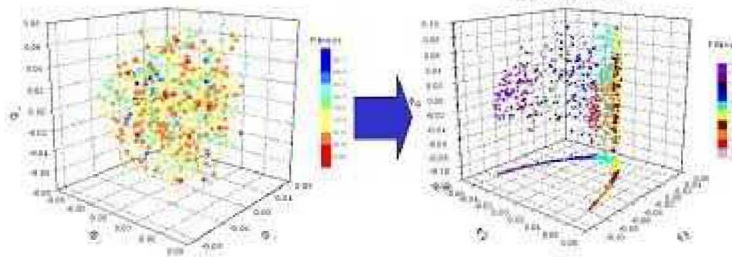
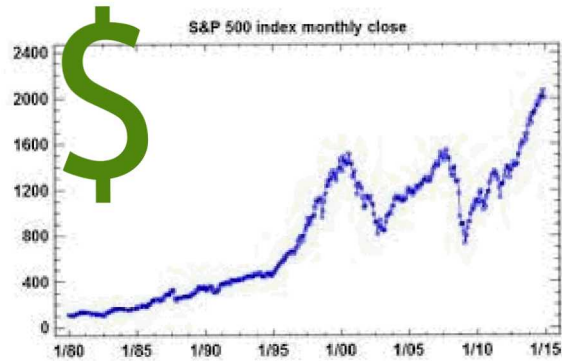


Can spiking really be used to solve non-cognitive tasks efficiently?



# Spiking circuits can efficiently solve stochastic differential equations

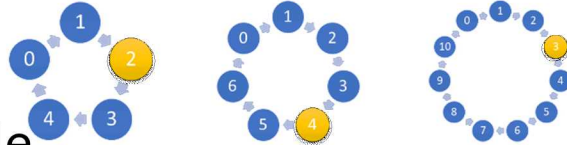
Diffusion: 
$$\frac{\partial C(x,t)}{\partial t} = D \frac{\partial^2 C(x,t)}{\partial x^2}$$



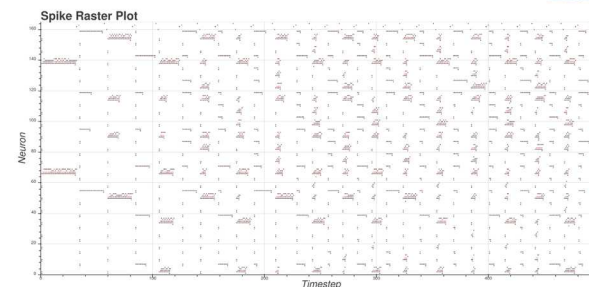
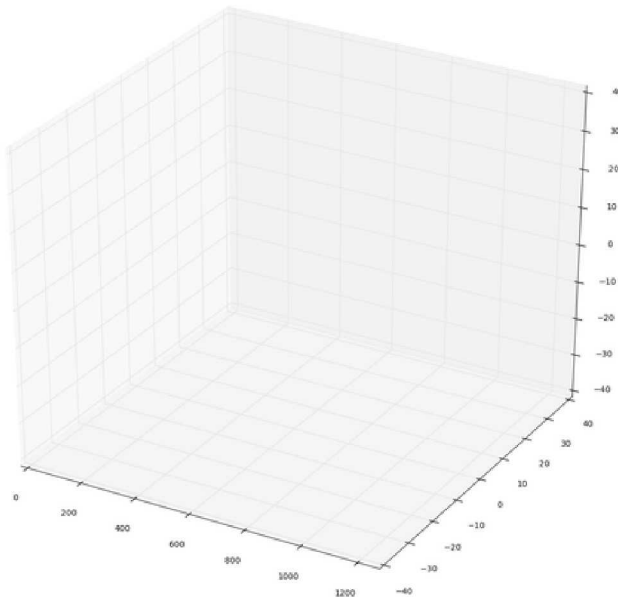
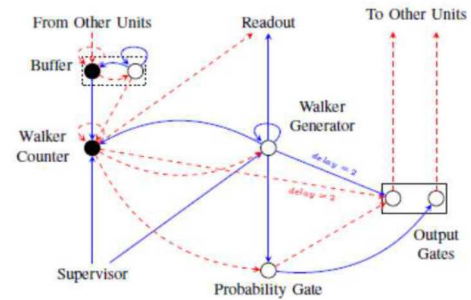
# Spiking circuits can efficiently solve stochastic differential equations

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

Modular circuit of  
spiking neurons  
per random walk particle



RW counting circuit of  
spiking neurons per  
simulation mesh vertex







A brief plug...



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

## NICE 2020

March 17-20<sup>th</sup>, 2020

Neuro-Inspired  
Computational Elements  
Workshop

Abstracts due  
November 1st, 2020



Im Neuenheimer Feld 227  
D-69120 Heidelberg  
Germany

Workshop: March 24-26<sup>th</sup> 2020  
Tutorials: March 27<sup>th</sup> 2020

## Heidelberg - Germany



Picture: fotolia.com / Sergey Borisov

## Kirchhoff Institute for Physics