SAND2018-1736C

Spiking Neuron Implementations of Several Fundamental Machine Learning Algorithms

Craig M. Vineyard, Stephen J. Verzi, William M. Severa, & James B. Aimone

1462—Data Driven and Neural Computing
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM

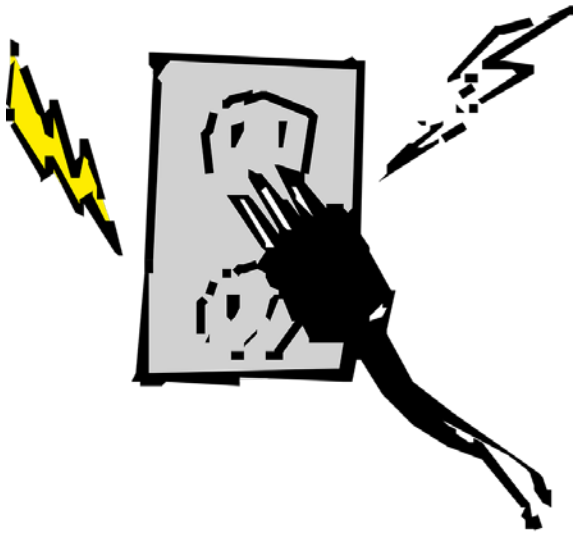
Sandia National Laboratories

Exceptional service in the national interest

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.

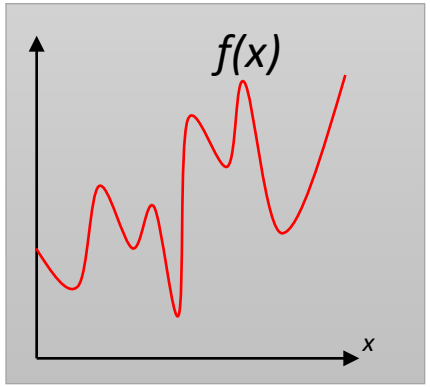
Introduction

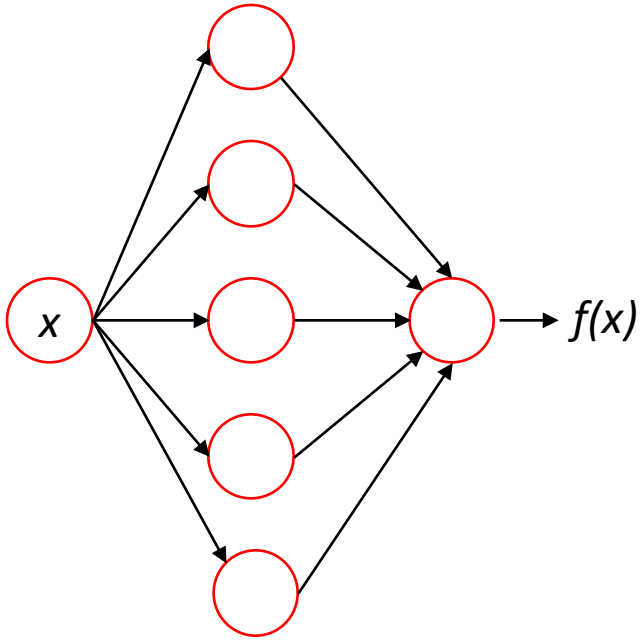
Google CEO Sundar Pichai said artificial intelligence “is one of the most important things humanity is working on. It’s more profound than, I don’t know, electricity or fire.” [MSNBC Interview January 2018]

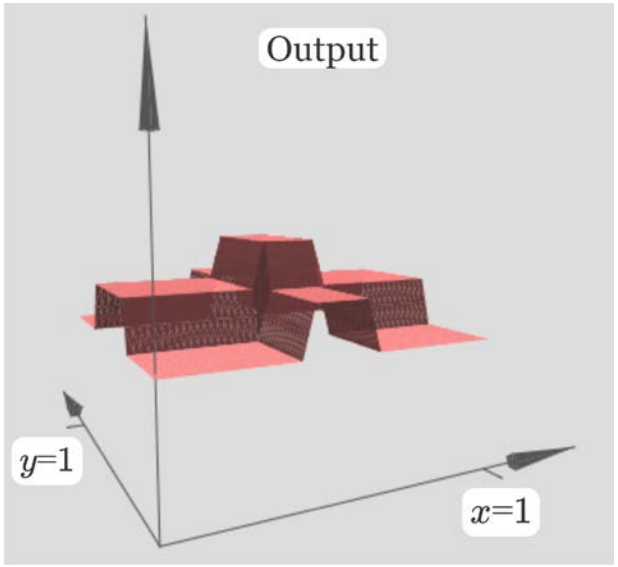


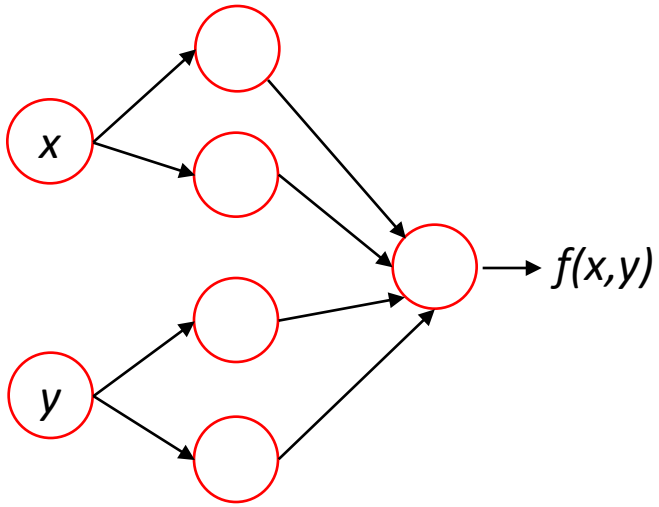
Can neural inspired computational elements deliver on this potential???

Universal Function Approximation



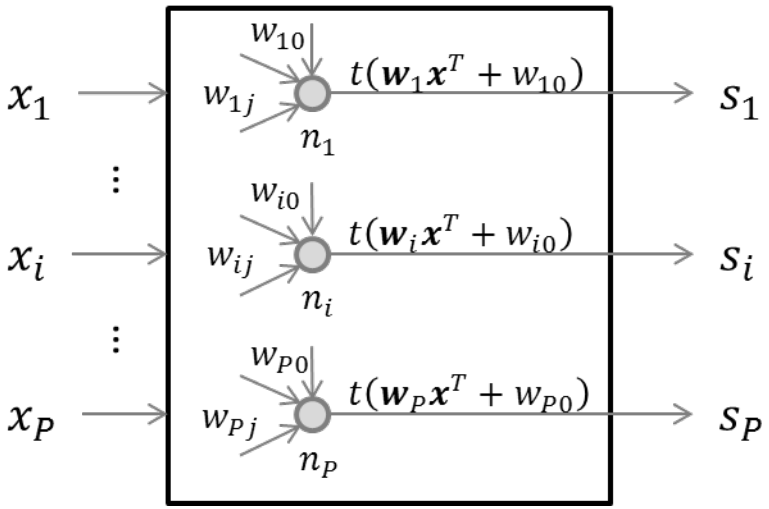




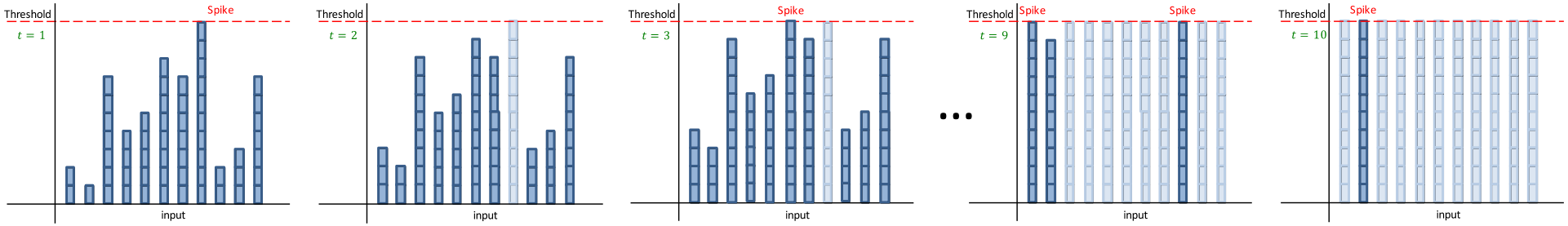


Neural Module

- Neurons n_i use discretized LIF
- Each neuron has a spiking threshold Θ (not shown in figure)
- Inputs to neurons are linear combinations of external input plus a bias signal
- Each neuron generates a temporal coded output s_i
- Each spike contains temporal coded information $t(\boldsymbol{w}_p * \boldsymbol{x}^T)$ which defines the latency of the spike signal



Fundamental algorithms using temporal coding



time-step	N = 1	2	3	4	5	6	7	8	9	10	11
1								1			
2											
3						1					
4			1				1				1
5											
6					1						
7				1							
8										1	
9	1								1		
10		1									

maximum

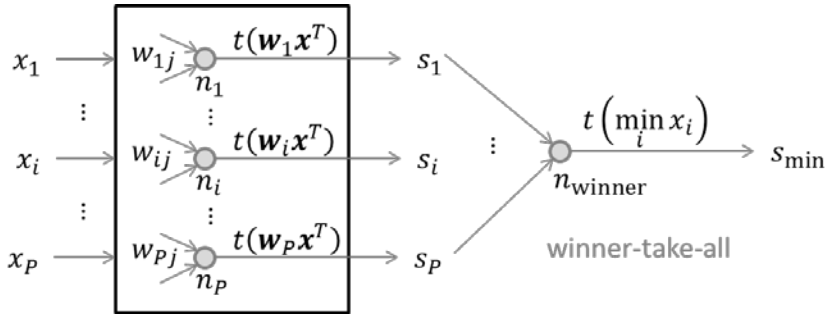
median

minimum

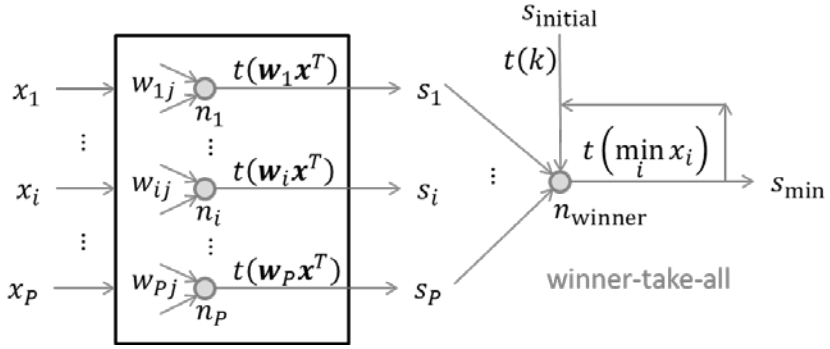
Sorted integers: $\{9, 7, 6, 6, 6, 5, 3, 2, 1, 1, 0\}$

SpikeMin

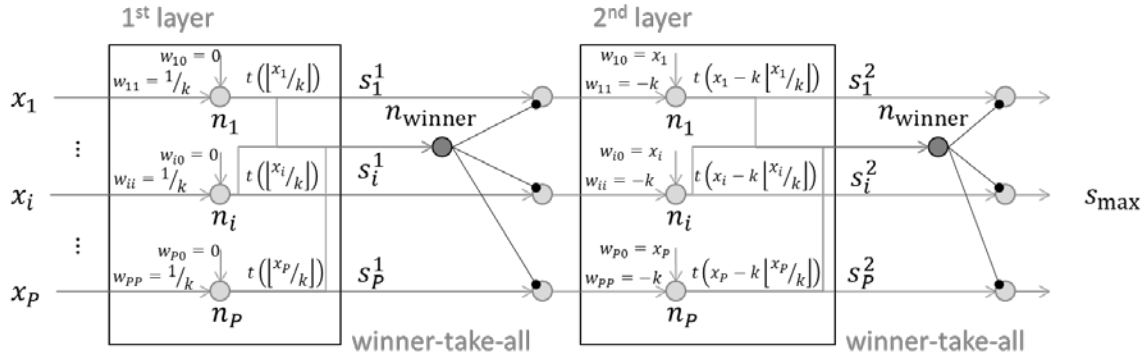
Finding the min where $P \geq N$



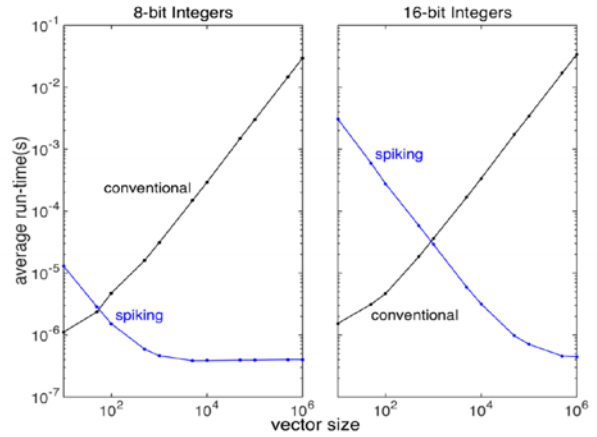
Finding the min where $P < N$



SpikeMax

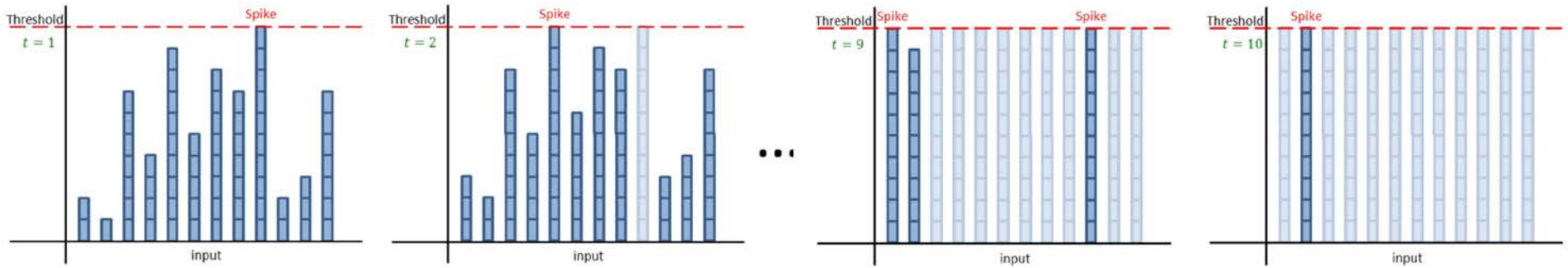


$$t\left(\max_i x_i\right) = ks_i^1 + s_i^2$$



Average runtimes for 10000 simulations of the spike-max neural spiking algorithm

Spiking Sort



Algorithm 1 spiking-sort

Input: set of integers, $\{x_1, x_2, \ldots, x_P\}$; k ▷ largest possible integer is $k-1$

Output: sparse bit matrix of spikes, S

$w = 0$ ▷ initialize weight matrix to all zeros

for $j \leftarrow 1$ to P, in parallel **do**

  $w_{0j} = 1$ ▷ initialize bias weights

  $\theta_j = k$ ▷ set neuron threshold

  $u_j = x_j$ ▷ directly inject initial value as neuron potential

$x_0 = 1$ ▷ initialize bias input

$\mathbf{S} = 0$ ▷ initialize bit matrix to all zeros

for $j \leftarrow 1$ to P, in parallel **do**

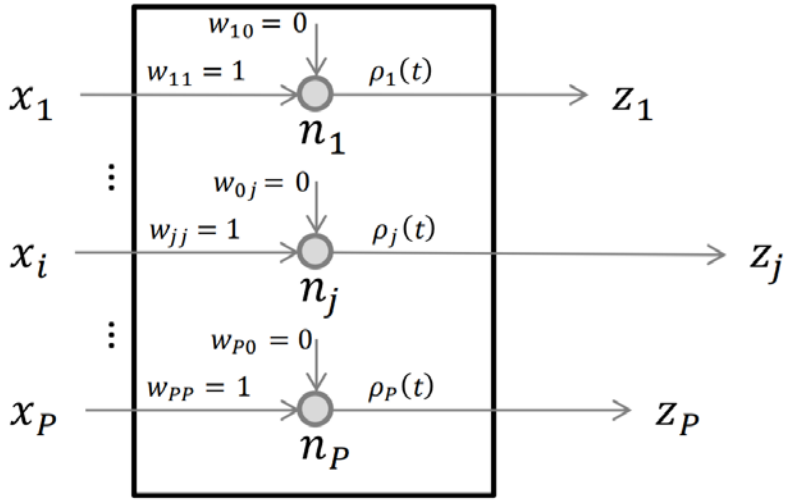
  for $\tau \leftarrow 1$ to k **do**

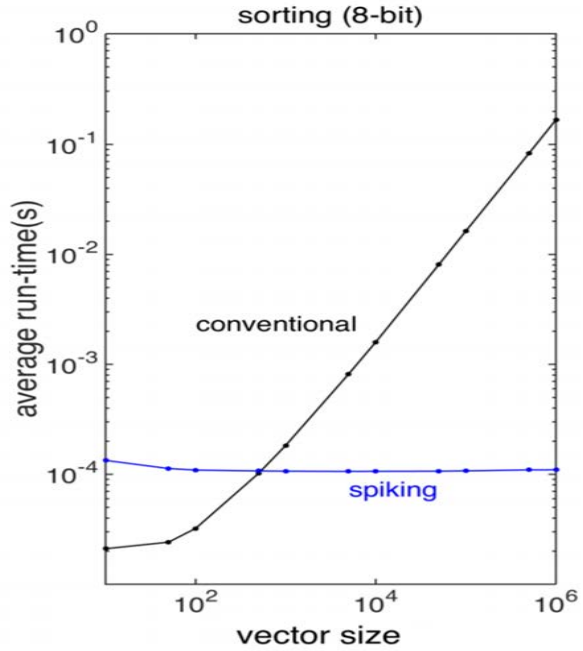
    $u_j = u_j + w_{0j}x_0$ ▷ neuron potential update (discretized LIF)

    if $u_j >= \theta_j$ **then** ▷ threshold check for spiking neuron

      $\mathbf{S}(\tau, j) = 1$

      $u_j = 0$ ▷ reset neuron potential after spike





Optimization Formula for the Median

- Given a set of floating point numbers $\mathrm{X} = \{x_1, x_2, \ldots, x_N\}$
- Compute the Signed Rank function

$$\underset{x \in \{x_i\}}{\tilde{R}(x)} = \sum_{i=1}^{N} \mathrm{sign}(x - x_i)$$

- The median, $\tilde{x}$, is such that $\tilde{R}(\tilde{x})$ is closest to 0

SpikeOpt(Median)

Sandia National Laboratories

Algorithm

Input: Set of integers, $\{x_1, x_2, \ldots, x_N\}$ where N is odd

Output: median integer, $m = \text{median}(x_i)$

typedef enum {INITIAL, SPIKING, DONE} is State

State $state \leftarrow SPIKING$ ▷ initialize state to SPIKING

for $i \leftarrow 1$ to N, in parallel **do**

$u_i = \sum_{j=1}^{N} \text{sign}(x_i - x_j)$

while $state \neq$ DONE **do**

if $u_i == 0$ **then**

$m = x_i$

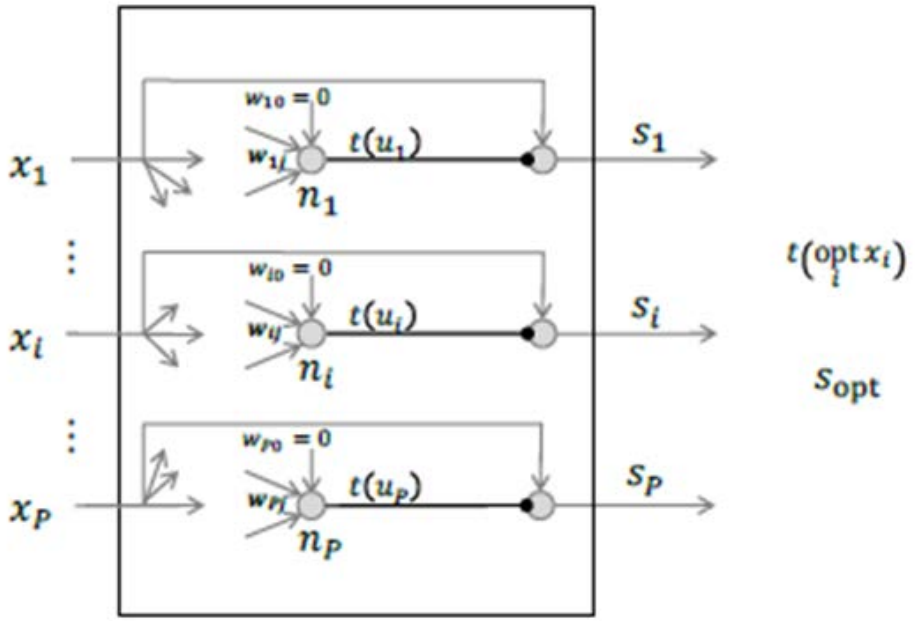
$state =$ DONE

else

$u_i = u_i - \text{sign}(u_i)$

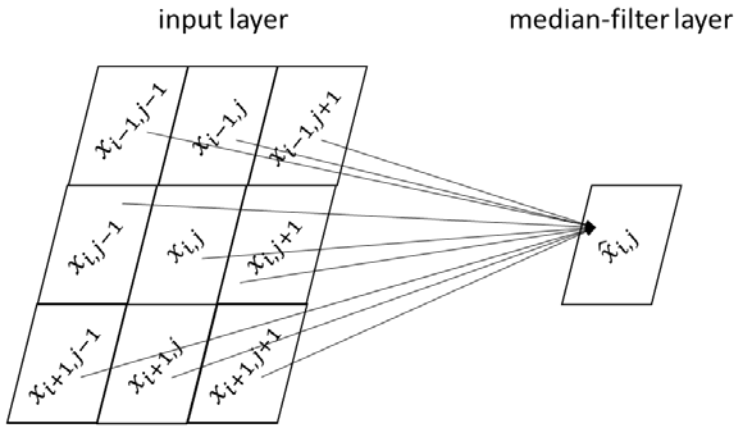
Architecture

Let $w_{ij} = \text{sign}(x_i - x_j)/x_j$



Median Filtering Example

- Median-filtering is an algorithm to perform noise reduction on an image or signal
- Run through image, pixel by pixel, and replace the current value us the value of the median of the neighbors
- Maximum size for each median operation is 9 which means we can we can compute the median filtered image in constant time using SpikeOpt(Median)



Original Image | Noisy Image | Filtered Image

Complexity Analysis

- Signed rank value will be in the range 0 to $\frac{N-1}{2}$
- Worst Case
  - SpikeOpt(Median) will operate for at most $\frac{N+1}{2}$ clock cycles
  - Total work $T_1 = O(N^2)$
  - Work per processor $T_P = O(N)$
  - Speedup $\frac{T_1}{T_P} = O(N)$
  - This is optimal when $P = N$
- Best Case
  - SpikeOpt(Median) will operate for at a minimum 1 clock cycle
  - Work per processor $T_P = O(1)$,

Complexity Analysis

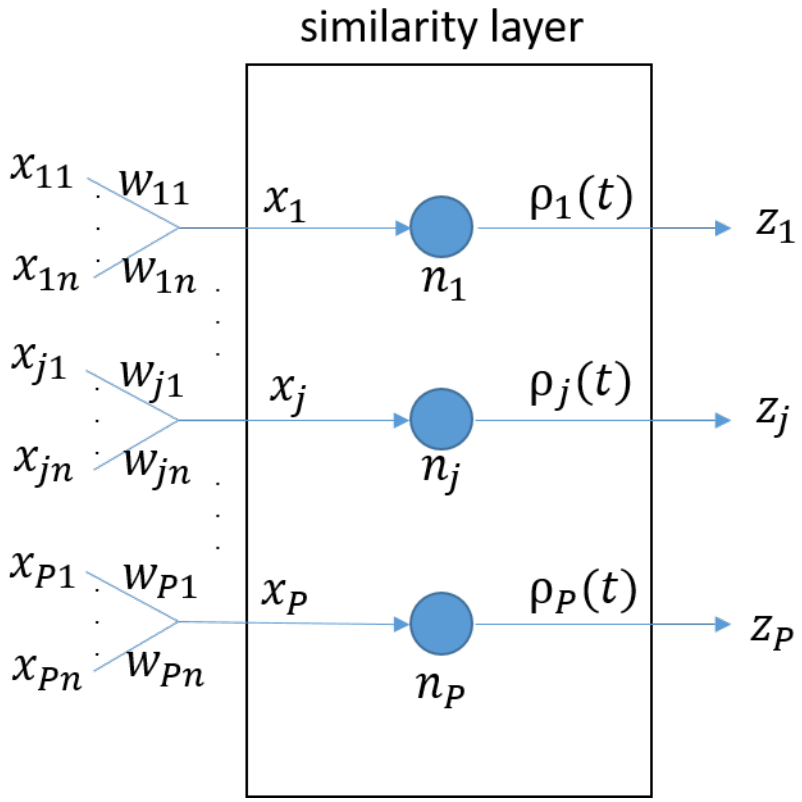
Theorem 1 – The SpikeOpt(median) algorithm achieves optimal runtime with the PRAM framework for a symmetric probability distribution

Theorem 2 - The SpikeOpt(median) algorithm achieves optimal runtime with the PRAM framework if each integer x_i is unique

Verzi, S. J., Vineyard, C. M., Vugrin, E. D., Galiardi, M., James, C. D., & Aimone, J. B. (2017, May). Optimization-based computation with spiking neurons. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (pp. 2015-2022). IEEE.

Spiking Similarity

- In many cases, machine learning algorithms are based upon a distance computation to infer relationships to other data points
- After the initial presentation of input values which are scaled and integrated then a nominal input is passed to all neurons driving them to fire
- This firing latency is inversely proportional to similarity between the input and the neural response encoding

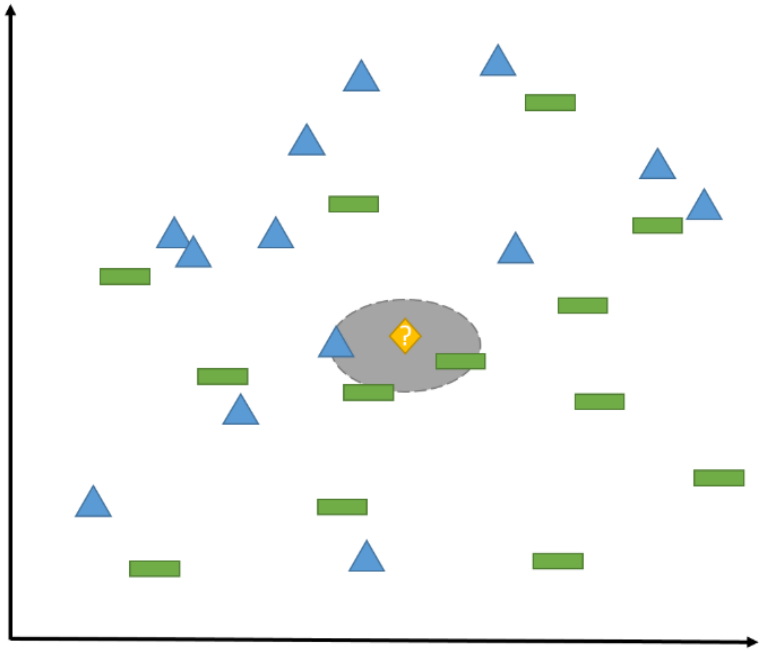


Nearest Neighbor

k-Nearest Neighbor (k-NN)

- Non-parametric method for classification

- Determines class membership as the class of the majority of the k nearest data points

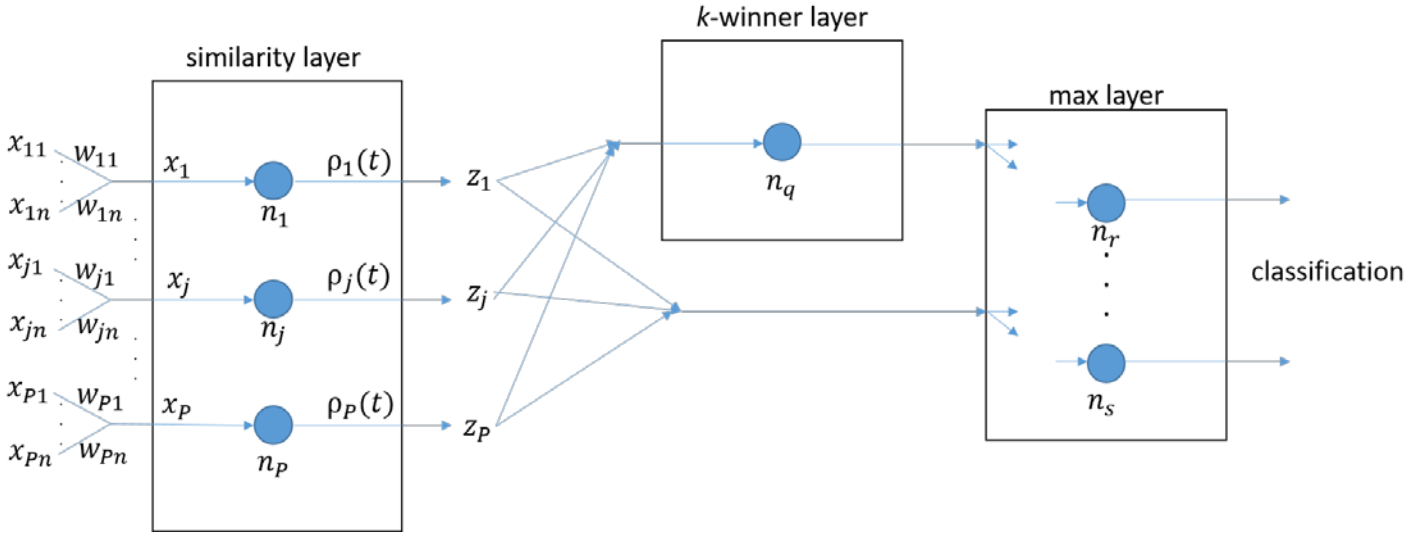


k-NN Algorithm:

Given query point *q*
Calculate distances from unknown point *q* to all data
Find k nearest neighbors
Vote on labels of k nearest neighbors

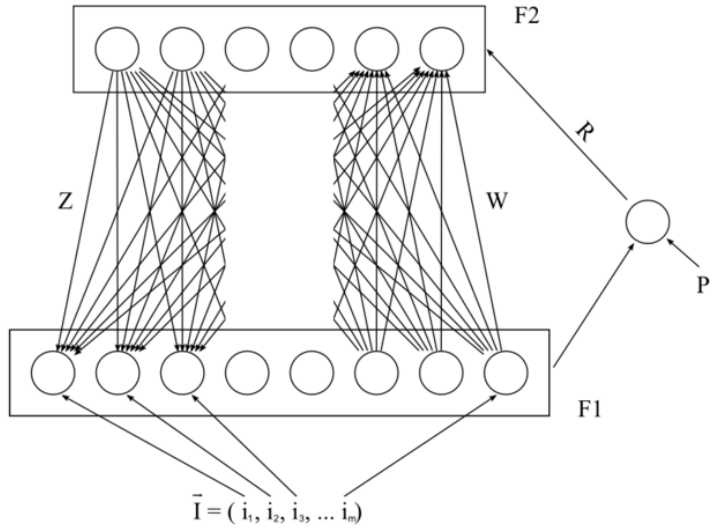
Spiking Nearest Neighbor (s-NN)

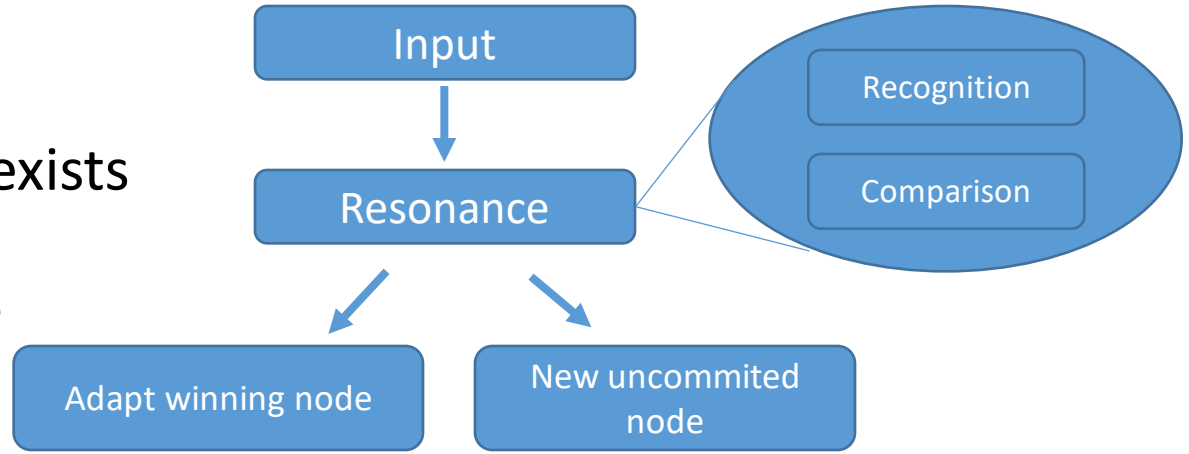
- Each neuron correspond to a data point
- Perform spiking similarity & identifying the first k spikes
- The k-winner layer determines when the k nearest neighbors have been found and then primes the max layer
- As a supervised problem, the classes attributed to the data are known & increment neurons corresponding to the individual classes



Adaptive Resonance Theory (ART)

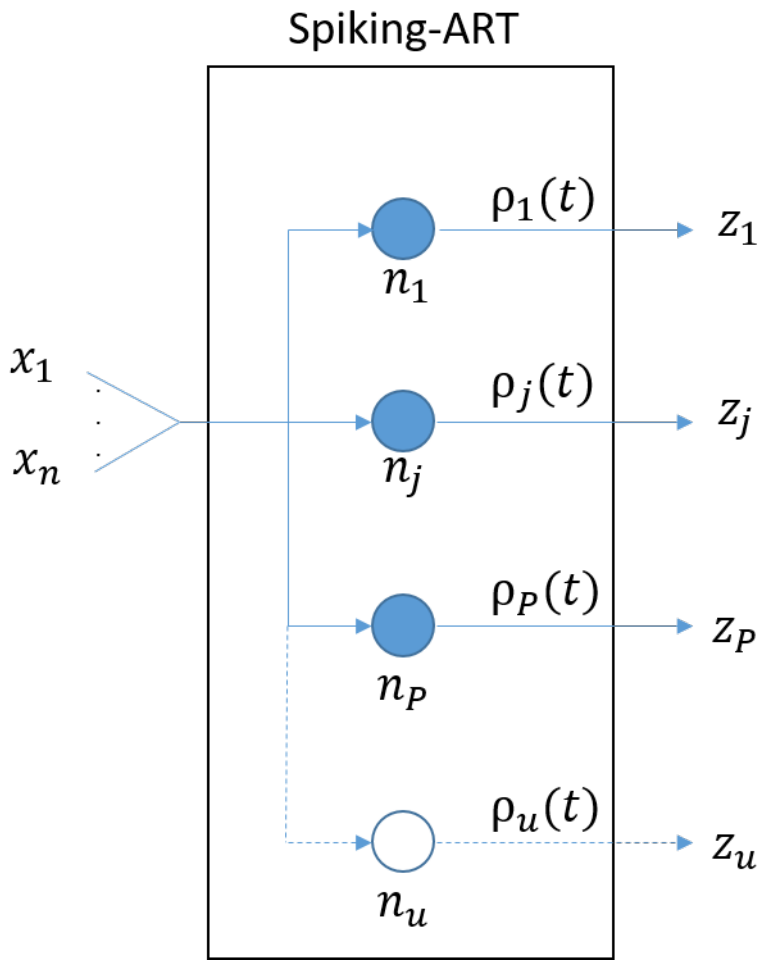
- Originally developed by Carpenter and Grossberg
- An online learning family of algorithms
- "Resonance" drives learning
  - Inputs are compared against stored templates
  - If a sufficiently similar representation exists
    - Update winning template
  - Otherwise a new category needs to be learned



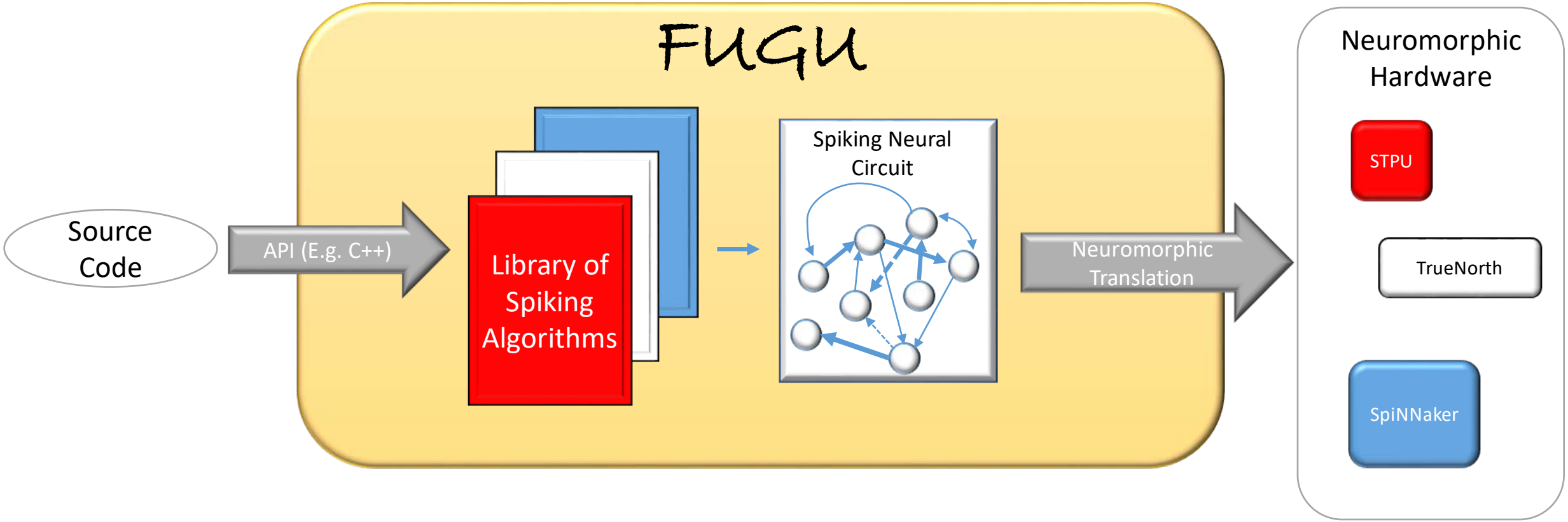


Spiking-ART

- Implemented by first performing spiking similarity to determine the closest matching template
- The vigilance similarity comparison constraint may be directly incorporated by only allowing a temporal response within ρ time steps
- If a sufficient match is found weights of the winning neuron are updated accordingly
- Otherwise a new uncommitted neuron is added with weights set to the present input



FUGU

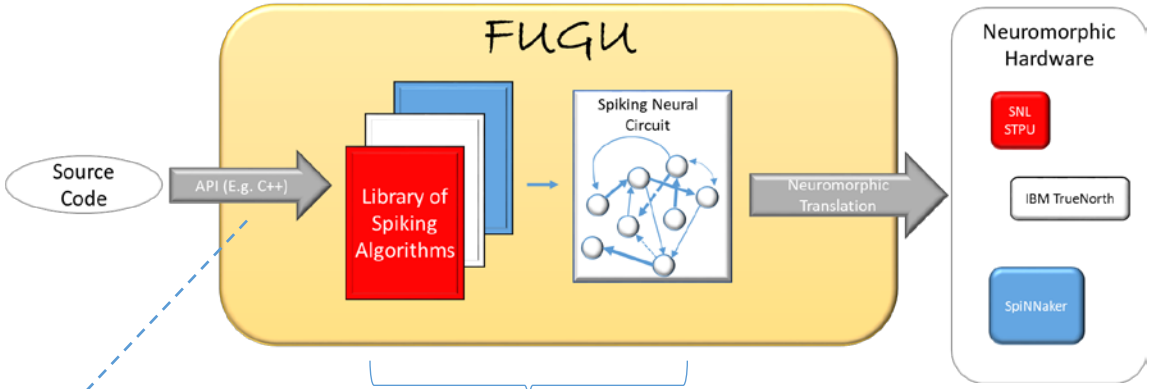

FUGU
Source Code
API (E.g. C++)
Library of Spiking Algorithms
Spiking Neural Circuit
Neuromorphic Translation
Neuromorphic Hardware
STPU
TrueNorth
SpiNNaker

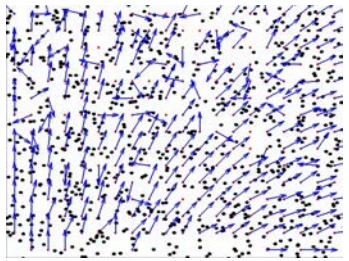
FUGU: PIV Cross-Correlation Example

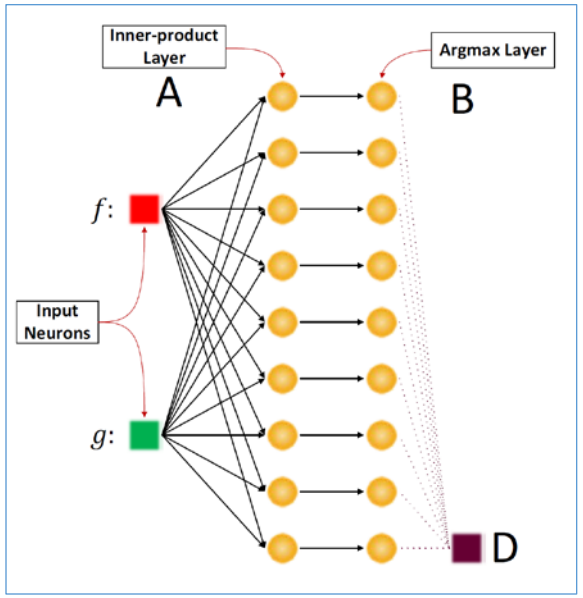
- Particle Image Velocimetry (PIV) is a well studied method for using particles to determine the local velocity flow in many applications throughout science and engineering

- Cross-Correlation finds agreement in signals
  - Computed as a sliding scalar product
  - $(f \star g)(n) = \Sigma_m f(n) g(m+n)$

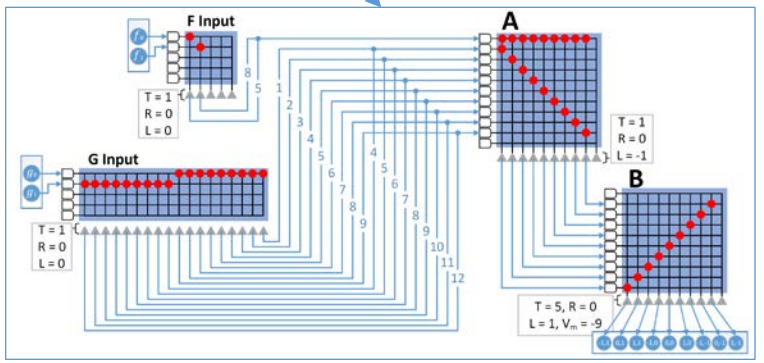
- Mapped to the SNL STPU & IBM TrueNorth Neuromorphic architectures







Severa et al.,ICRC2016



Conclusion

- There are some bold & exciting claims surrounding neuromorphic computing
- Presented spiking neural circuit implementations of several fundamental computer science & machine learning algorithms
  - Working on neuromorphic implementation as well as computational complexity analysis
  - Broad applicability and various benefits
- Just scratching the surface of NICE potential – looking forward to the amazing algorithms & architectures over next few days!

Sandia National Laboratories

Neuromorphic Hardware in Practice and Use

Description of the workshop

- Abstract – This workshop is designed to explore the current advances, challenges and best practices for working with and implementing algorithms on neuromorphic hardware. Despite growing availability of prominent biologically inspired architectures and corresponding interest, practical guidelines and results are scattered and disparate. This leads to wasted repeated effort and poor exposure of state-of-the-art results. We collect cutting edge results from a variety of application spaces providing both an up-to-date, in-depth discussion for domain experts as well as an accessible starting point for newcomers.

Goals & Objectives

- This workshop strives to bring together algorithm and architecture researchers and help facilitate how challenges each face can be overcome for mutual benefit. In particular, by focusing on neuromorphic hardware practice and use, an emphasis on understanding the strengths and weaknesses of these emerging approaches can help to identify and convey the significance of research developments. This overarching goal is intended to be addressed by the following workshop objectives:
  - Explore implemented or otherwise real-world usage of neuromorphic hardware platforms
  - Help develop 'best practices' for developing neuromorphic-ready algorithms and software
  - Bridge the gap between hardware design and theoretical algorithms
  - Begin to establish formal benchmarks to understand the significance and impact of neuromorphic architectures

http://neuroscience.sandia.gov/research/wcci2018.html

Call: https://easychair.org/cfp/nipu2018