University of Pittsburgh

Algorithm Innovations of Enhancing Scalability and Adaptability of Learning Systems

Yiran Chen
Evolutionary Intelligence Lab (EI-Lab)
University of Pittsburgh

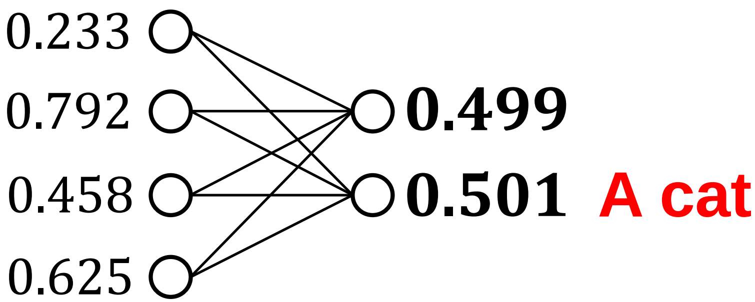
Neuromorphic Computing Workshop 2016
@Oak Ridge National Laboratory

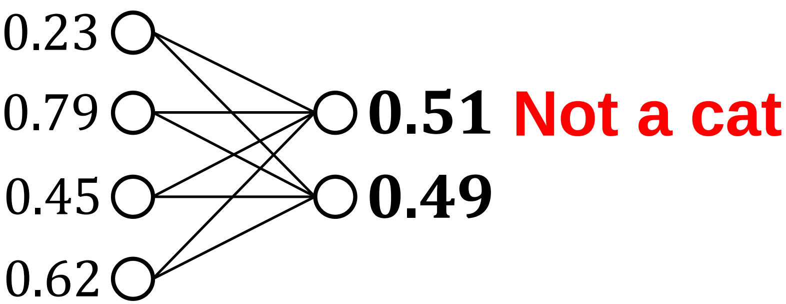
Mismatch: Hardware vs. Software

	Hardware	Software
Model/Component scale	Small/Moderate	Large
Re-configurability	Hard	Easy
Accuracy vs. Power	Tradeoff	Accuracy
Training implementation	Hard	Easy
Precision vs. Limited programmability	Low precision (often a few bits)	Double (high) precision
Connectivity realization	Hard	Easy

Precision & Limited programmability

Decrease precision without re-training



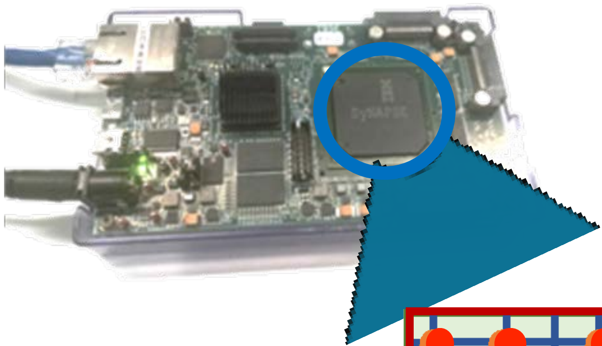


Re-train low precision AlexNet on ImageNet

Precision	Top-5 Accuracy
32-bit floating point	80.3%
16-bit floating point	80.3%
8-bit fixed point	80.1%
4-bit fixed point	**14.0%**
2-bit fixed point	**0.9%**

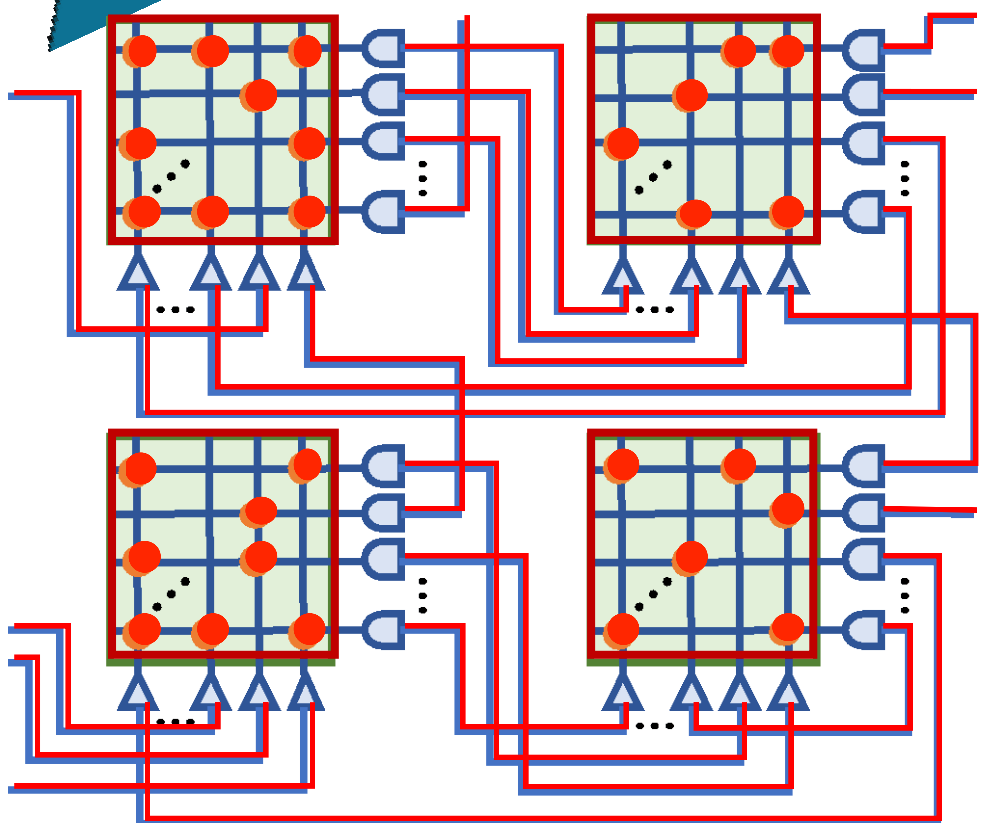
Example: TrueNorth

TrueNorth Chipset Architecture



The IBM TureNorth Dev Board.*   * *A. S. Cassidy, etc, SC14*

- 4,096 neurosynaptic cores;
- 1 million neurons;
- 256 million synapses;
- A 65mW real-time neurosynaptic processor。



A Network of Neurosynaptic Cores

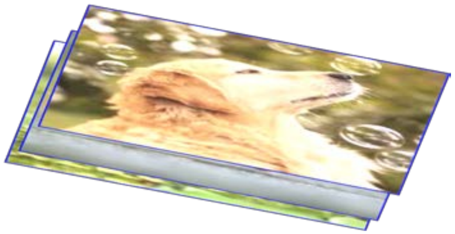
A 256×256 Synaptic Crossbar

Low-resolution Integer Weight

Spike Communication

Overview of TrueNorth Operation

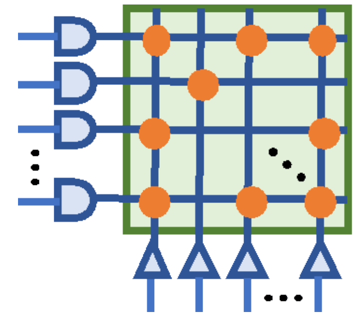
Database



Raw Pixels
Testing

Spike Encoder

Binary Spikes



TrueNorth model with low resolution synaptic integers

Raw Pixels
Training

Tea learning in Caffe (Traditional CPUs/GPUs)

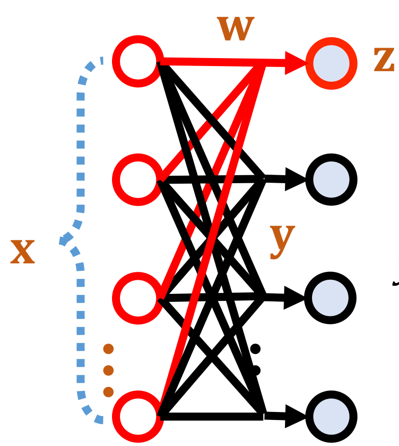
A network with floating-point weights

Tea deploying in CPE (Corelet Programming Environment)

* *S. K. Esser, etc., NIPS'15*

Learning and Deploying of TrueNorth

Mapping Neural Networks in IBM TrueNorth

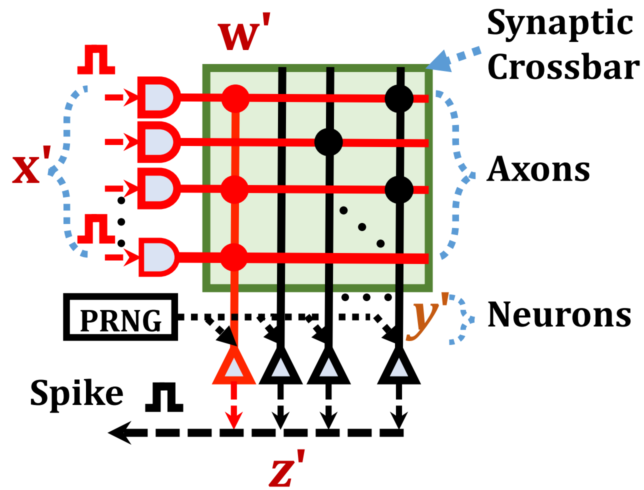


Floating-point Precision

$y = w \cdot x + b$

$z = h(y)$

Traditional Neural Networks



Neural Networks with TrueNorth

Binary/low Integer precision

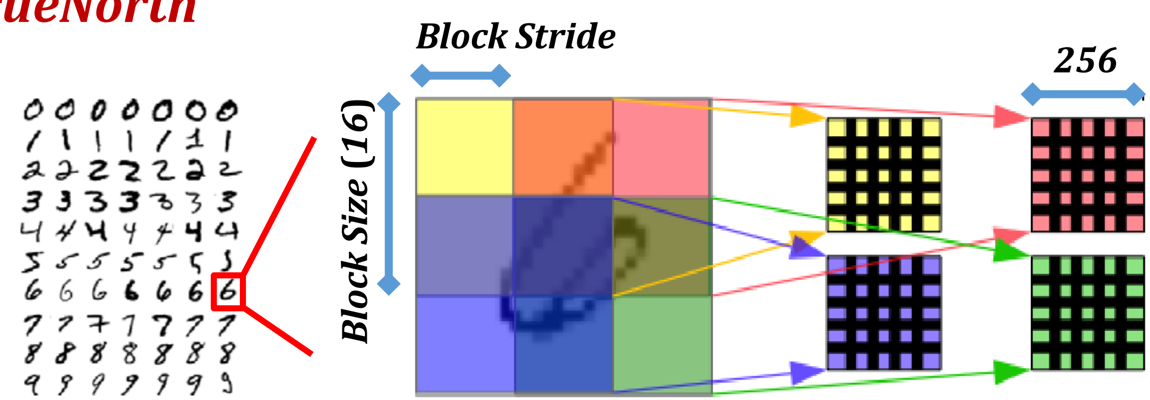
McCulloch-Pitts neuron model:

$$y' = w' \cdot x' - \lambda$$

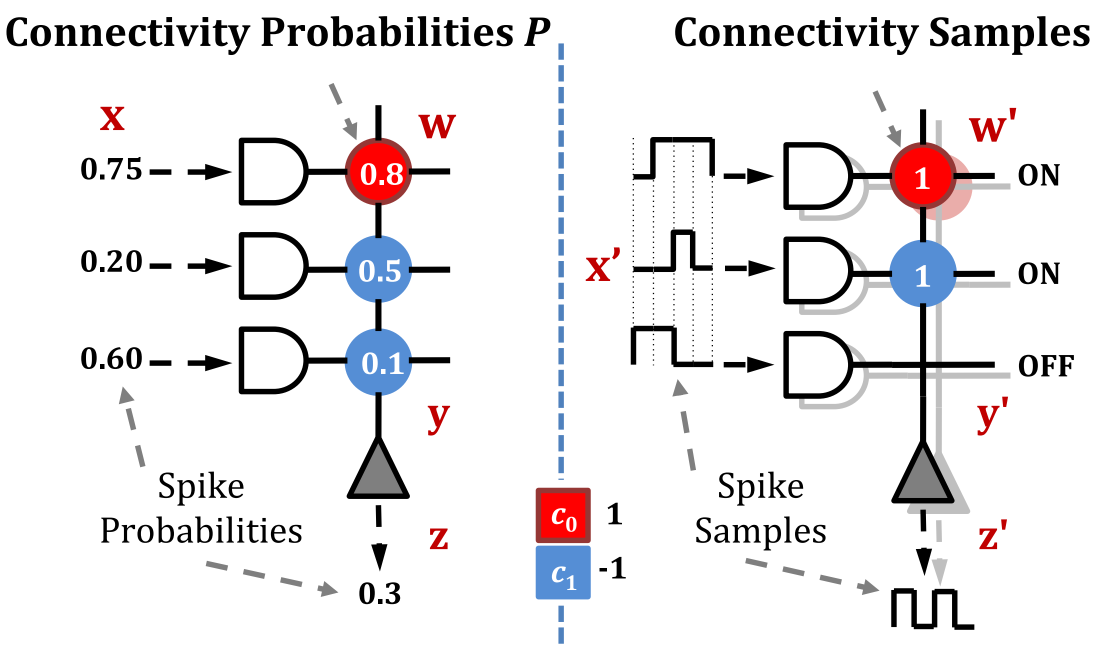
$$z' = \begin{cases} 1, & \textit{Reset } y'=0; \textit{ If } y' >= 0. \\ 0, & \textit{Reset } y'=0; \textit{ If } y' < 0. \end{cases}$$

MNIST with TrueNorth

MNIST database



Learning and Deploying of TrueNorth



(a) Tea learning
Traditional Floating Point Precision

(b) Tea deploying
Binary/low integer precision
sampled by float-point probability

$$p_i = \frac{w_i}{c^{(i)}}, c^{(i)} \in \{c_0, c_1\}$$

$$\begin{cases} P(w'_i = c^{(i)}) = p_i, \\ P(w'_i = 0) = 1 - p_i \end{cases}$$

$$\begin{cases} P(x'_i = 1) = x_i, \\ P(x'_i = 0) = 1 - x_i \end{cases}$$

$$y' = \sum_{i=0}^{n-1} w'_i x'_i$$

$$\begin{aligned} E\{y'\} &= E\left\{\sum_{i=0}^{n-1} w'_i x'_i\right\} \\ &= \textstyle\sum_{i=0}^{n-1} E\{w'_i\}E\{x'_i\} \\ &= \textstyle\sum_{i=0}^{n-1} p_i c^{(i)} x_i\} = \sum_{i=0}^{n-1} x_i w_i \\ &= y \end{aligned}$$

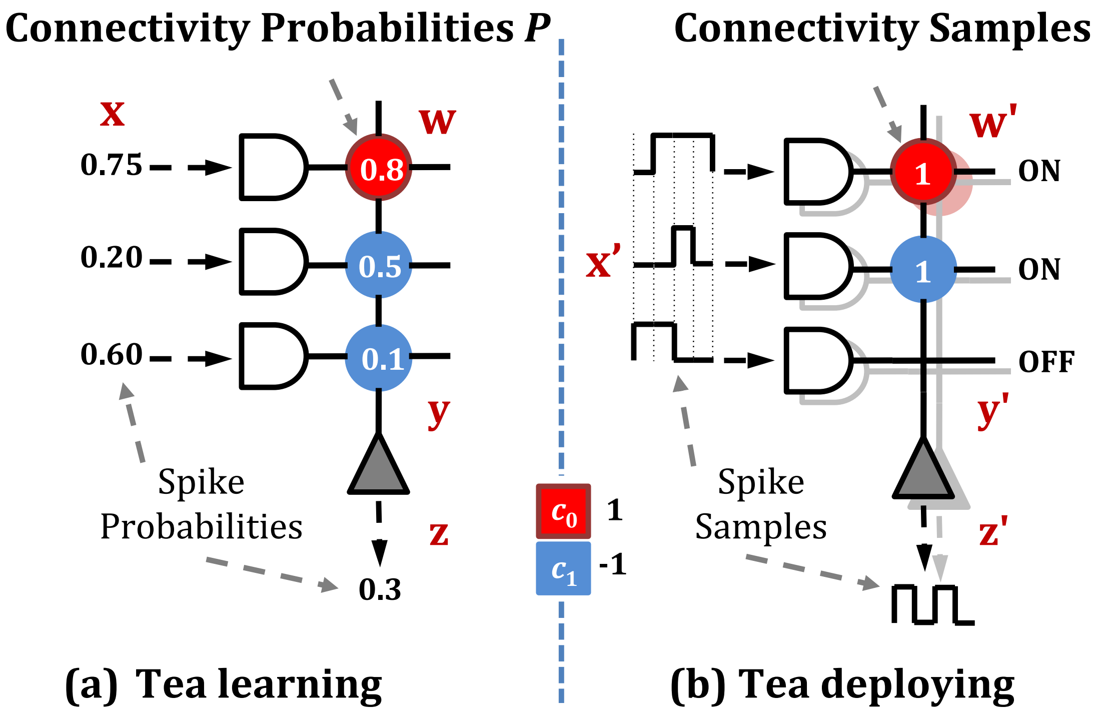
$$\begin{aligned} E\{z'\} &= P(y' \geq 0) \\ &= \frac{1}{2}\left[1 + erf\left(\frac{-\mu_{y'}}{\sqrt{2}\sigma_{y'}}\right)\right] \end{aligned}$$

MNIST Accuracy:

- 95.27% in Caffe
- 90.04% @1 NN copy & 1 *spf* in TrueNorth
- 92.74% @ 1 NN copy & 4 *spf* in TrueNorth
- 94.63% @16 NN copies & 1 *spf* in TrueNorth

(NN - neural networks; *spf* – spikes per frame)

Learning and Deploying of TrueNorth



(a) Tea learning (b) Tea deploying

$$\Delta y = y' - y$$
$$= \sum_{i=0}^{n-1} w'_i x'_i - \sum_{i=0}^{n-1} w_i x_i$$

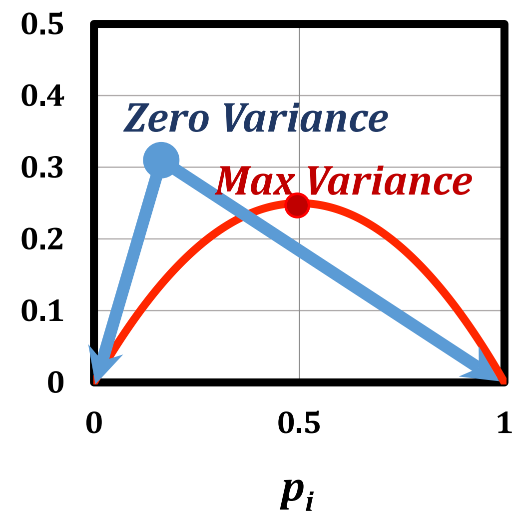
$$E\{\Delta y\} = 0$$
$$var\{\Delta y\} = \sum_{i=0}^{n-1} var\{w_i^{'} x_i^{'}\}$$

Unbiased approximation w/ variance is affected by both synaptic and spiking randomness.

Spiking Randomness:
Determined by External Data

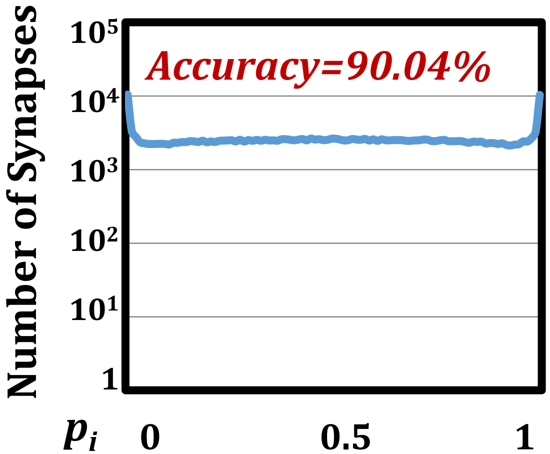
Synaptic Randomness:

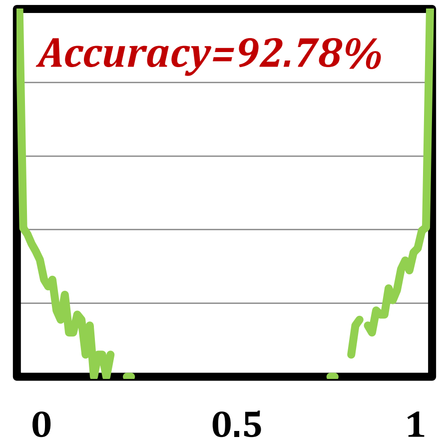
$$var\{w_i^{'}\} = E\left\{\left(w_i^{'}\right)^2\right\} - E\{w_i^{'}\}^2 = p_i(1-p_i)$$



Minimizing Deployment Variance

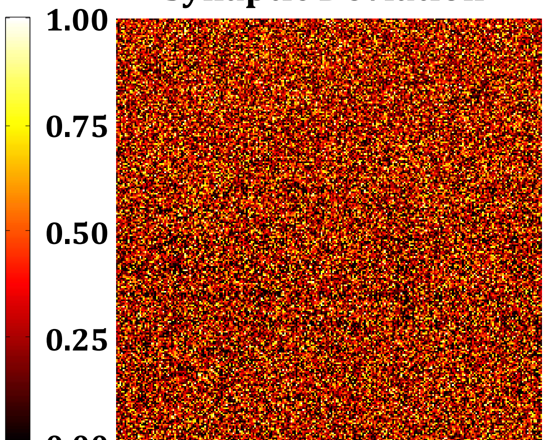
Minimization target: $\hat{E}(w) = E_D(w) + \boxed{\lambda \times E_p(P)}$ Data Less **Probability Regulation**





$$E_{p_1}(P) = \| |P - a| - b \| = \sum_{i=1}^{M} |p_i - a| - b$$

Synaptic Deviation

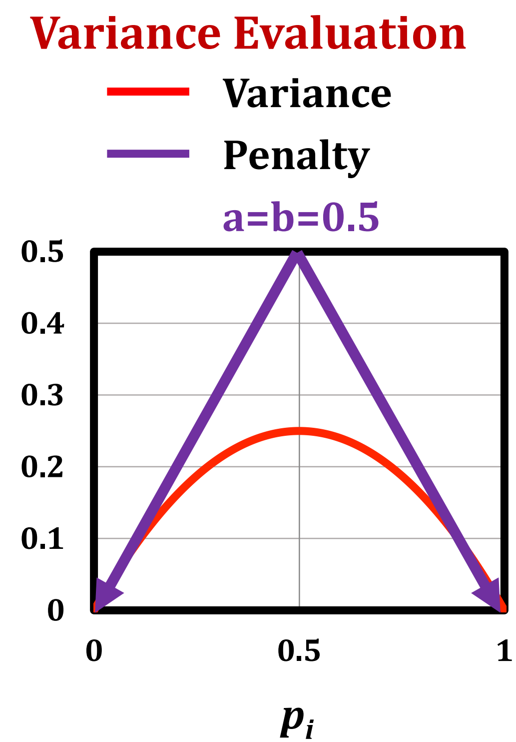


Baseline

@ 1 Network copy & 1 spf

Synaptic Deviation

Probability Regularization



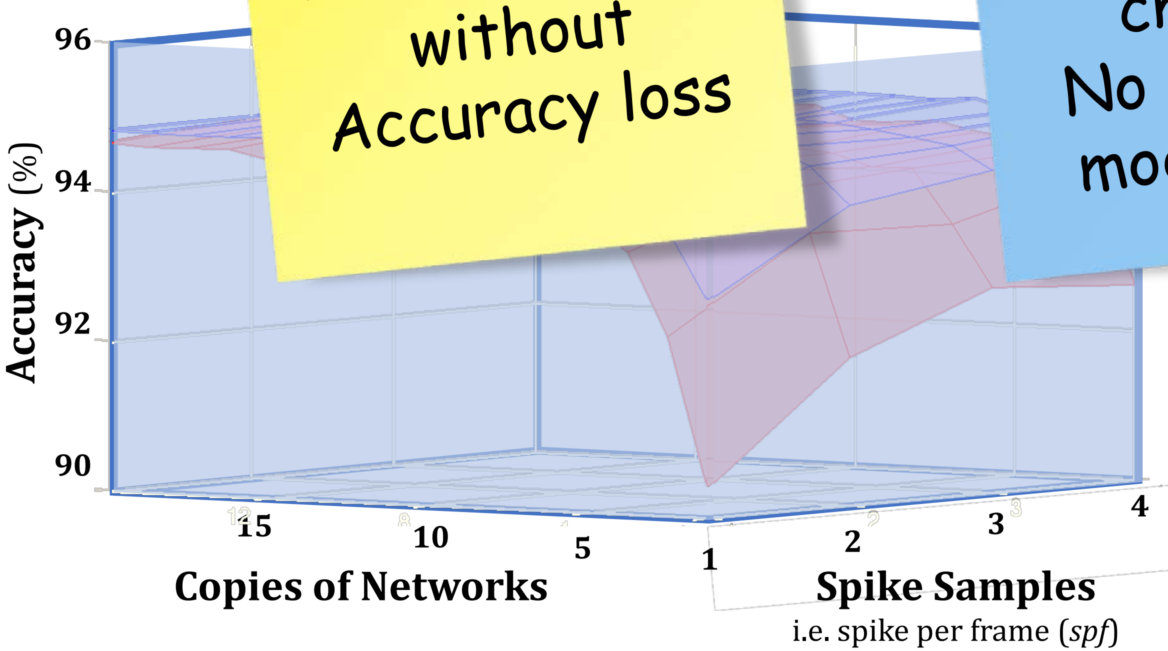
Experiment Results

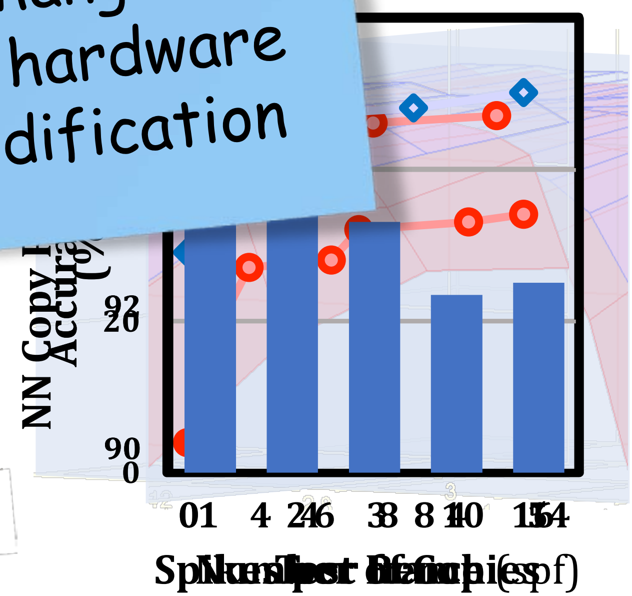
Test Bench	Dataset	Block stride	Hidden Layer #	Cores per Layer	Accuracy in Caffe
1	MNIST	12	1	4	95.27%

* 16 and 9 correspond to the cores utilized by 1st and 2nd hidden layer.

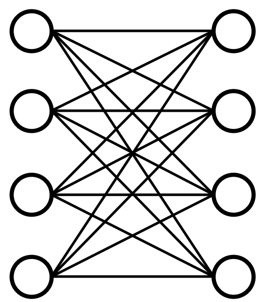
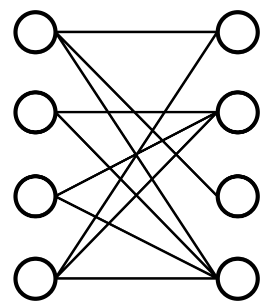
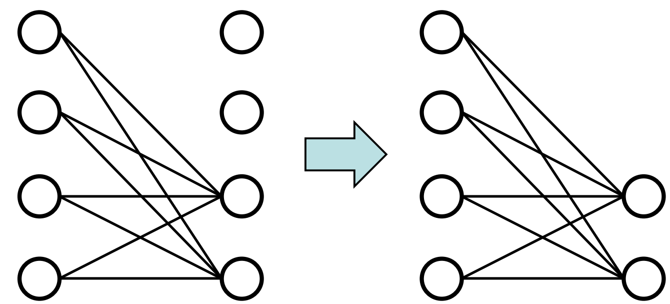
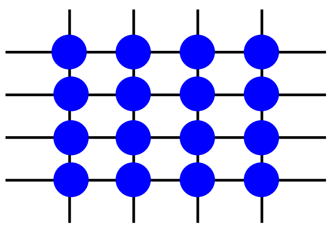
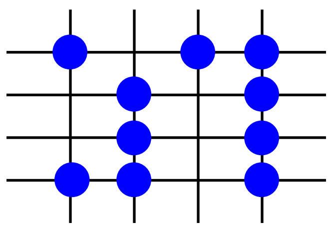
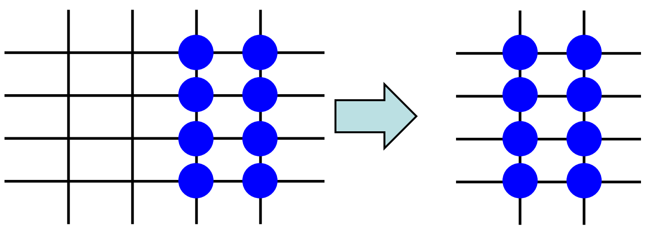
33.4% Avg. Core Reduction without Accuracy loss

Only Tea learning is changed. No hardware modification





Sparsity & Computation Cost

	Dense	Random sparsity	Structured sparsity
Network			
Deployed weight matrix			
Memory occupancy	Full & Large	Ineffective & Large	Full & Small
Power	High	Still high	Low

Messages

- Neuromorphic engineering, also known as **neuromorphic computing**, is a concept developed by Carver Mead in the late 1980s, describing the use of **very-large-scale integration (VLSI) systems** containing electronic analog circuits to mimic **neuro-biological** architectures present in the nervous system. (wikipedia.org)
  - Hardware (computing) and Software (bio-model and algorithm)
- "Chicken and Egg"
  - Another hardware-software co-design problem?
  - Harder than that as we do not have a solid theory and implementation foundation.
- Coordination and Standardization are needed in research of neuromorphic computing.

Sponsors

Yiran Chen

EI-lab