



BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Analog Memory-Based Techniques for Accelerating Deep Neural Networks

Hsinyu (Sidney) Tsai
IBM Almaden Research Center – San Jose, CA USA

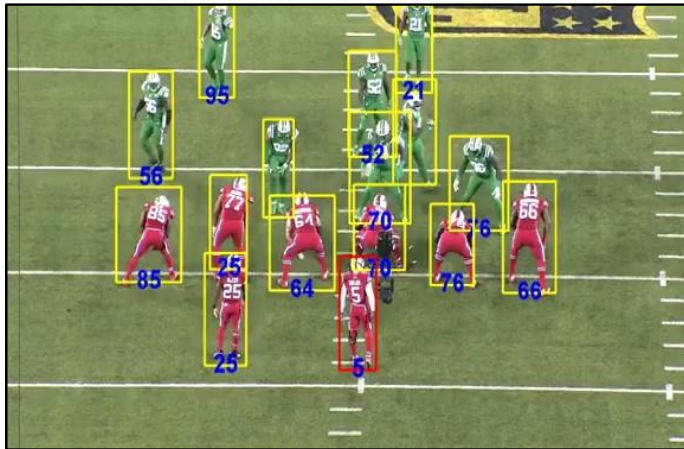
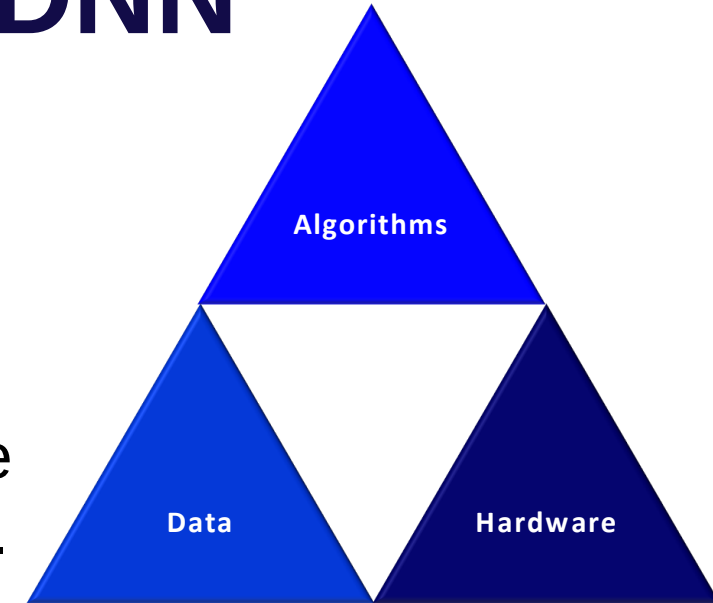


Outline

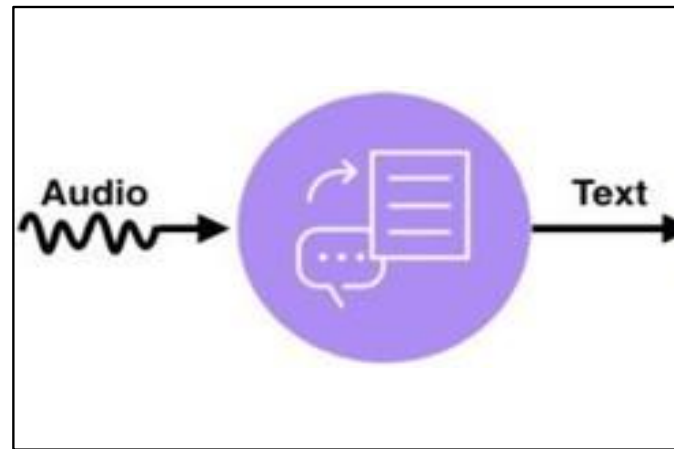
- Introduction – Deep Neural Networks (DNN) and Analog Memory
- Phase Change Memory for DNN Training and Inference
- Energy Efficiency for Analog Memory-Based Techniques
- Summary

The Rise of AI and DNN

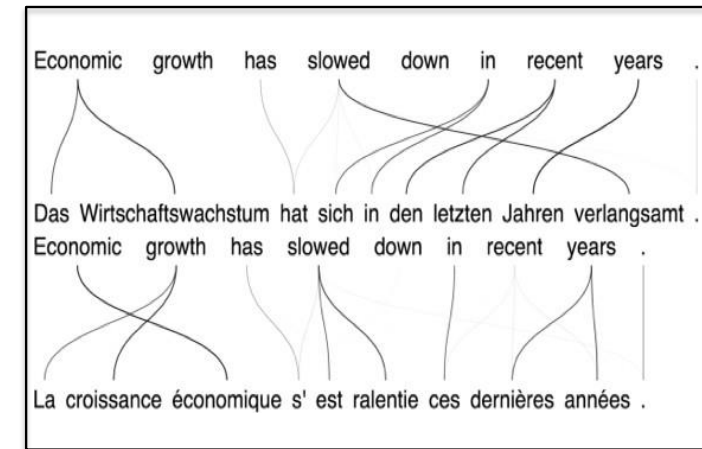
- The rise of AI relied on improving algorithms, abundant data, and accelerating hardware.
- Deep Neural Network (DNN), as a major field of AI, has surpassed human-level accuracy in some tasks and outperformed most rule-based models.



Object Detection

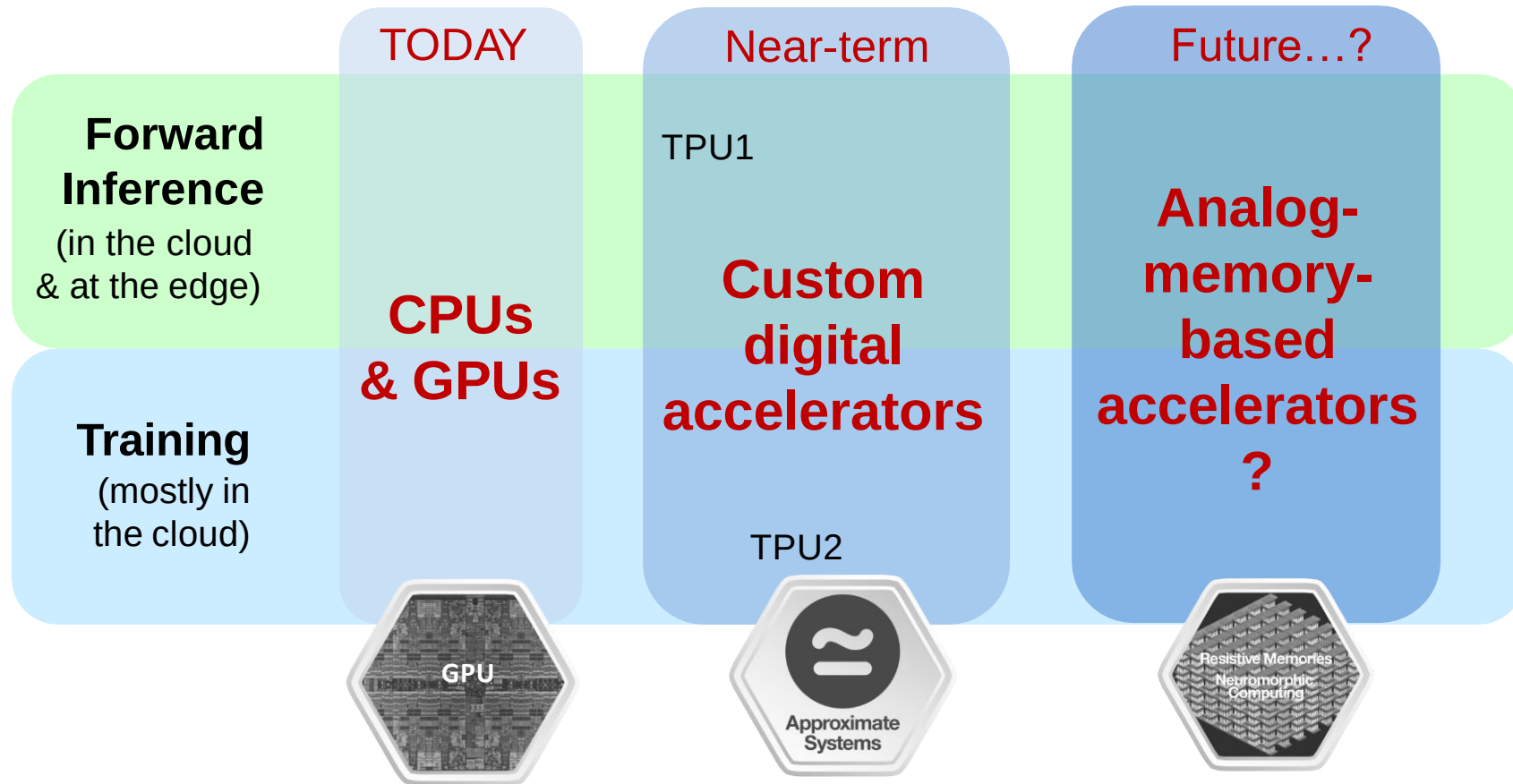


Speech Transcription



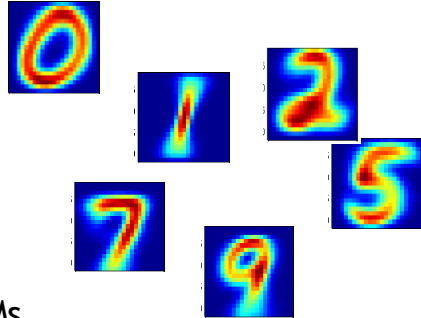
Language Translation

AI Hardware: from Today to the Future

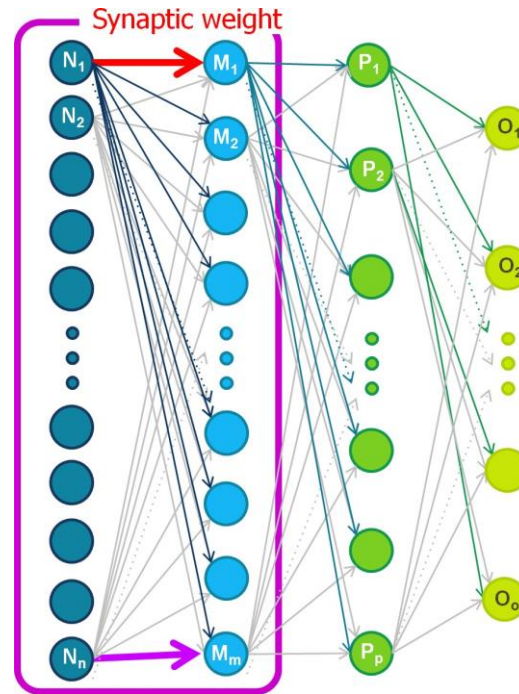


Deep Neural Network (DNN)

Input data (images,
raw speech data, etc.)
input to neural network

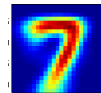


“MNIST” database
~1998
→ check-reading ATMs



A Deep Neural Network contains
multiple **layers**, ...
each layer containing many **neurons**, ...
each neuron driven through many
synaptic **weight** connections
from other neurons.

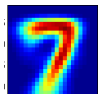
Training:



UN-trained network

“um.. I have no idea?”

Forward inference:



Fully trained network

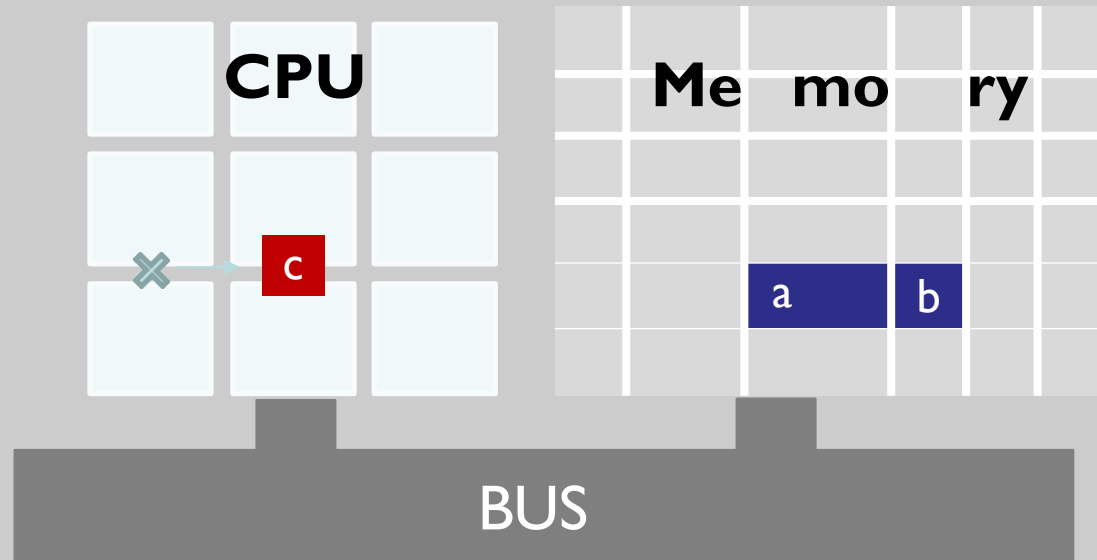
“This is a **seven**.”

“This is a seven.”

Hardware opportunity: Efficient, **low-power** deployment

Data Movement Cost

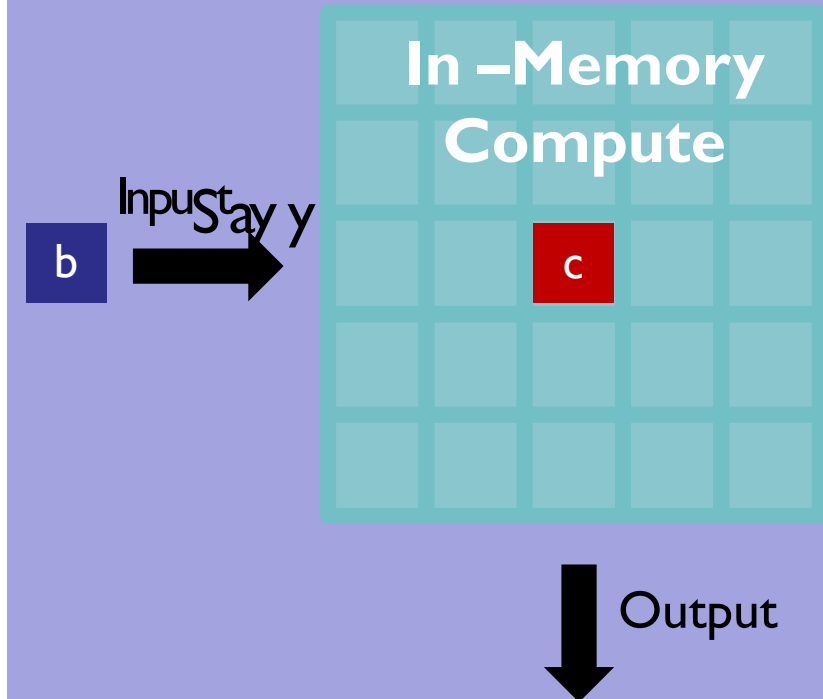
Digital Accelerators



Von Neumann Architecture

- ✗ Consumes high energy in data movement
- ✗ Bus has limited bandwidth & can be a bottleneck

Analog Accelerators

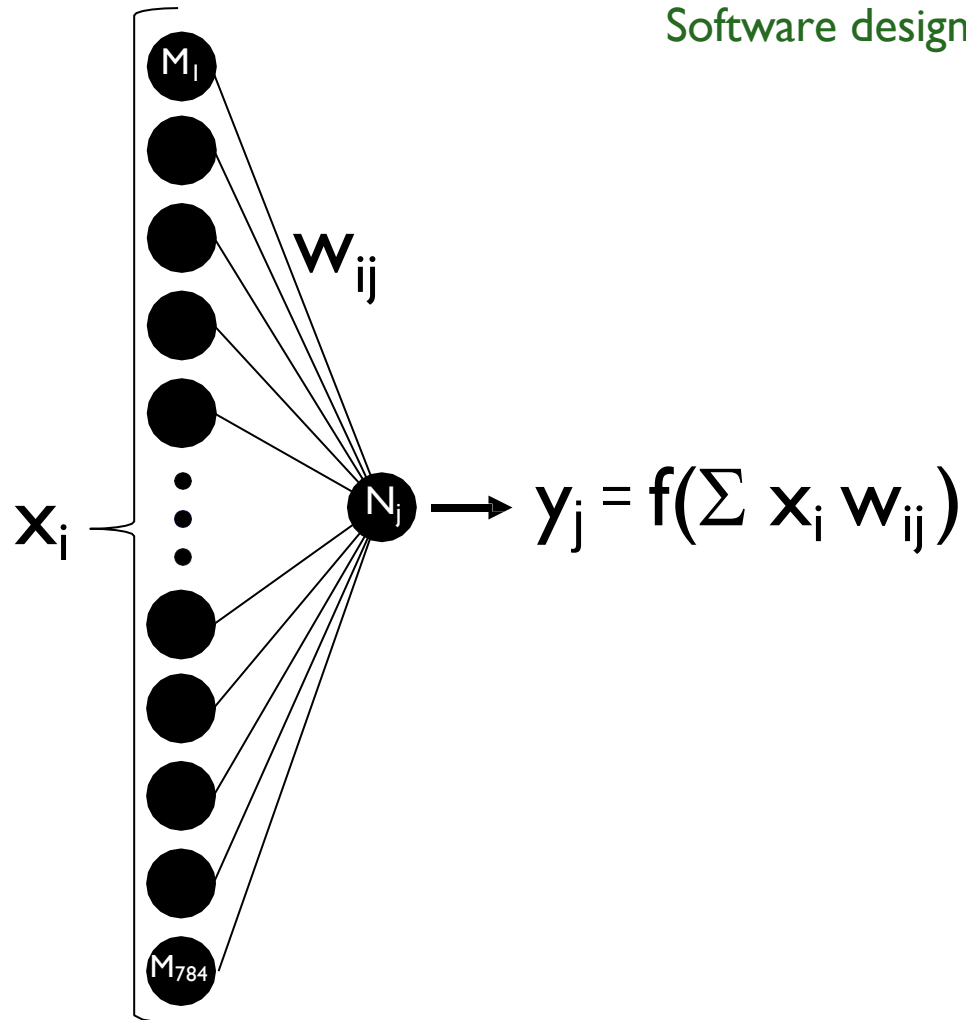


How is the network mapped into memory, and how is the multiplication performed?

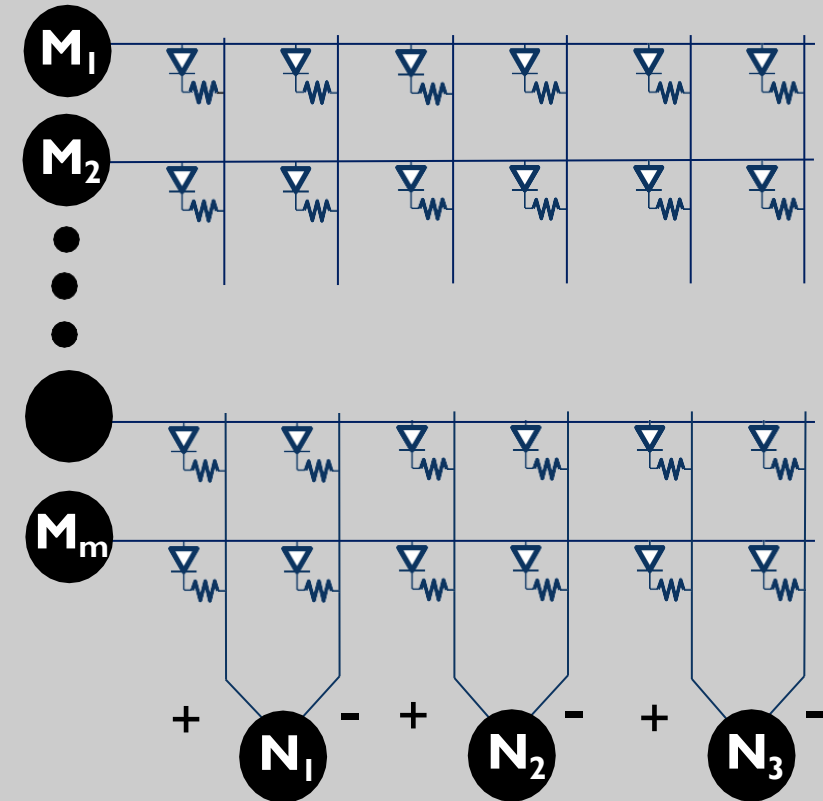
Analog in-memory computing offers better energy efficiency and throughput

Multiply-Accumulate with NVM

Software design



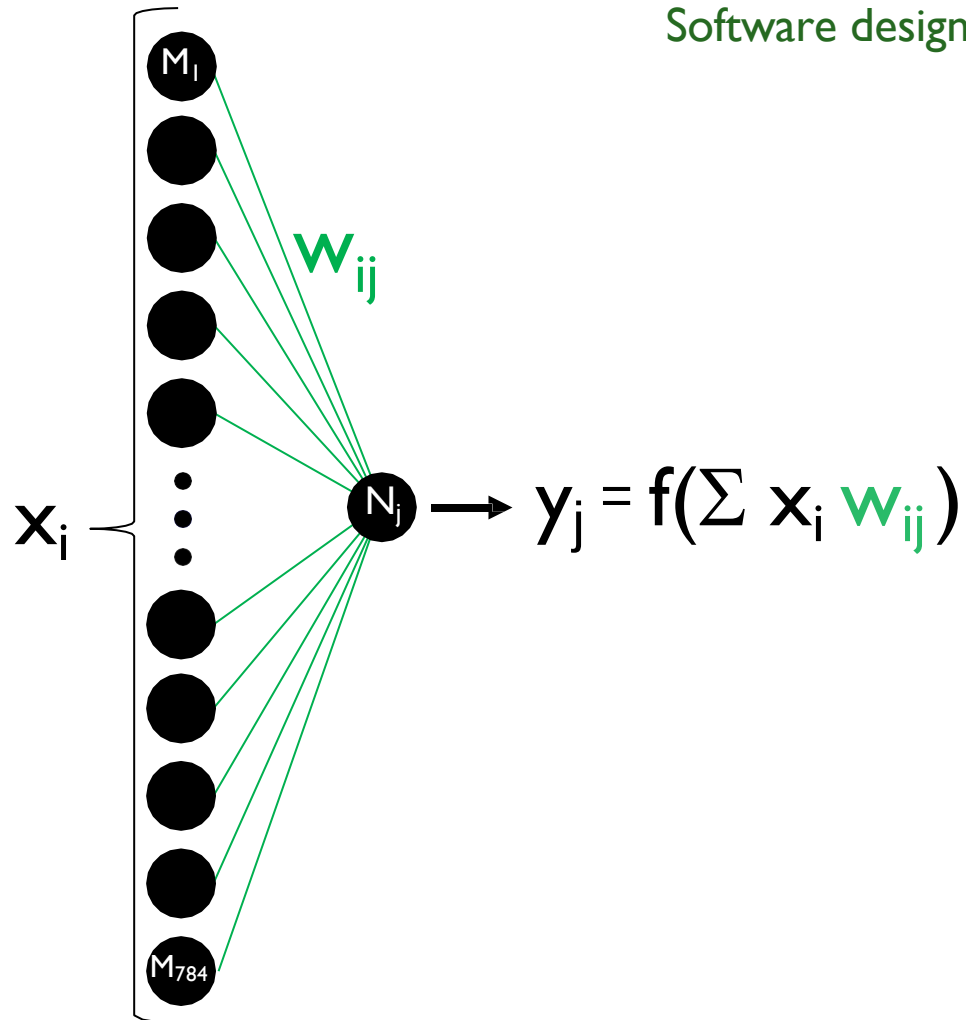
Hardware implementation



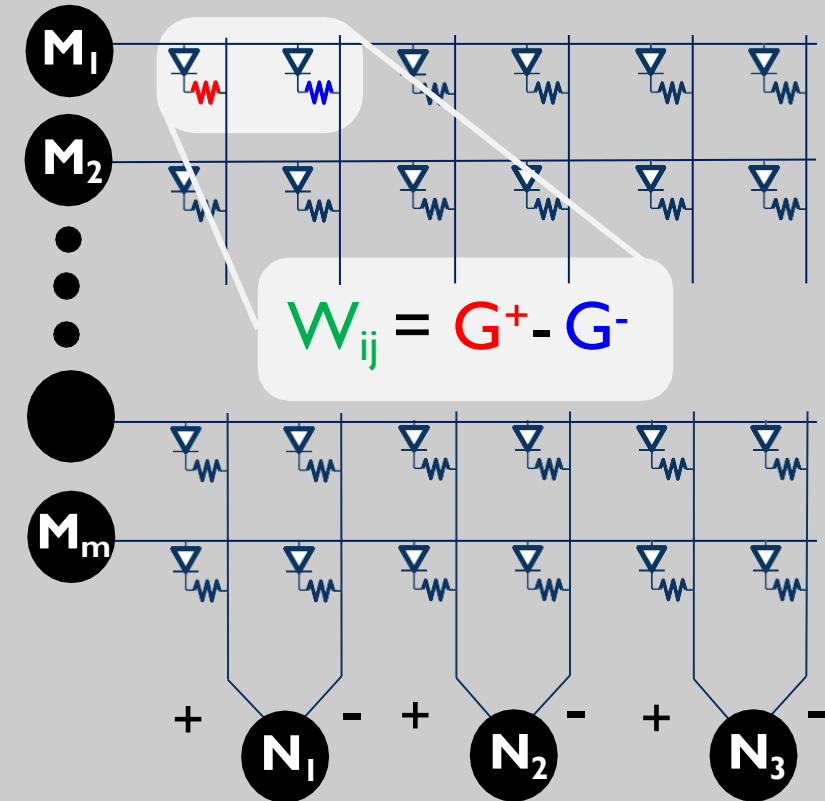
- How to map a neural network on a NVM crossbar array?

Multiply-Accumulate with NVM

Software design



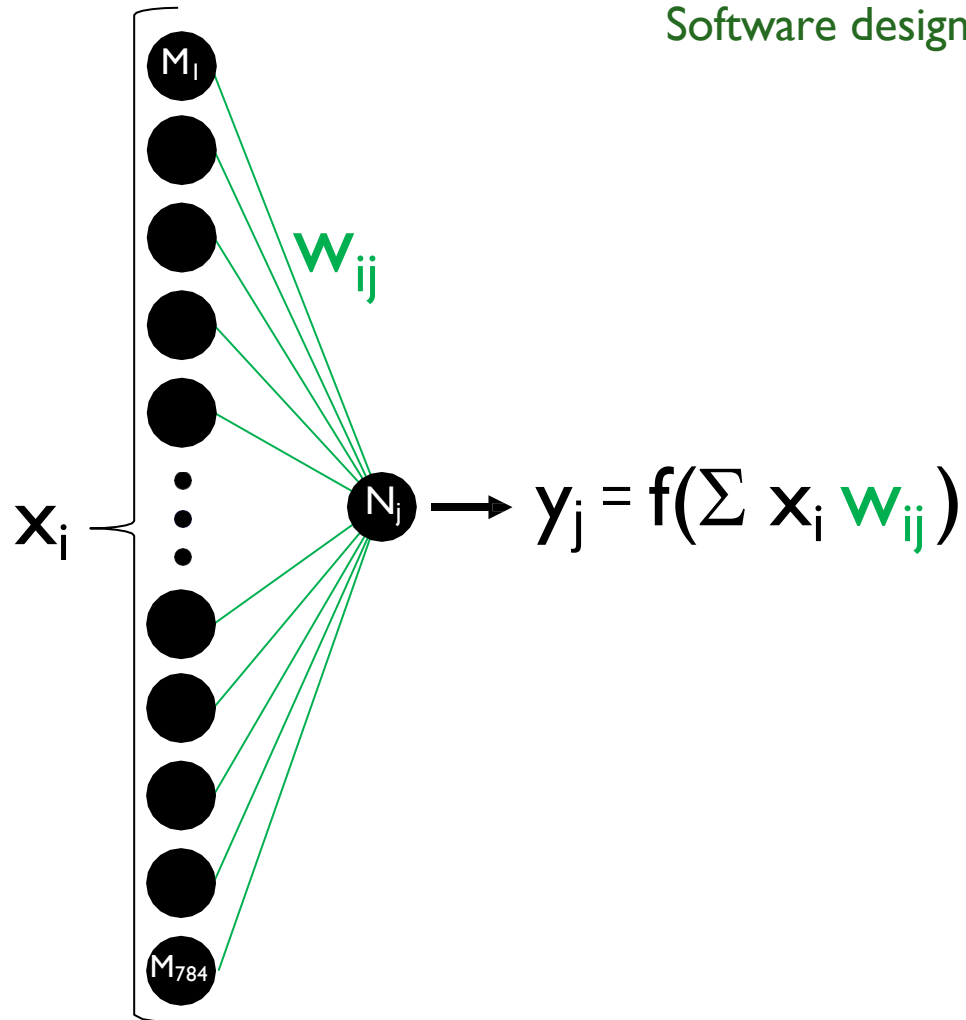
Hardware implementation



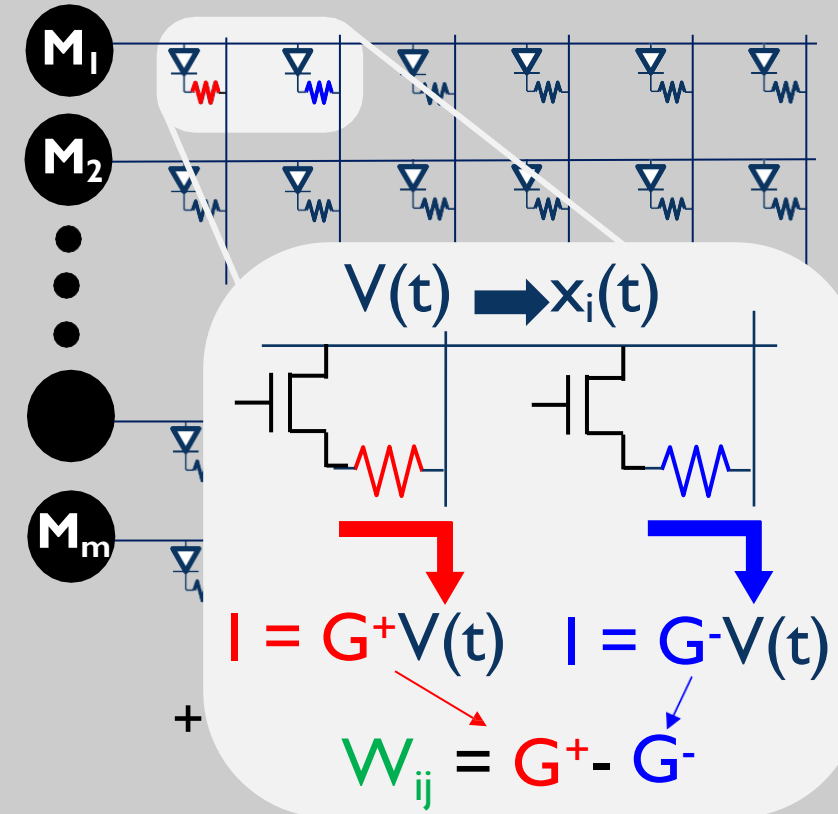
- The **weight** is encoded with a conductance pair (G^+ , G^-)

Multiply-Accumulate with NVM

Software design



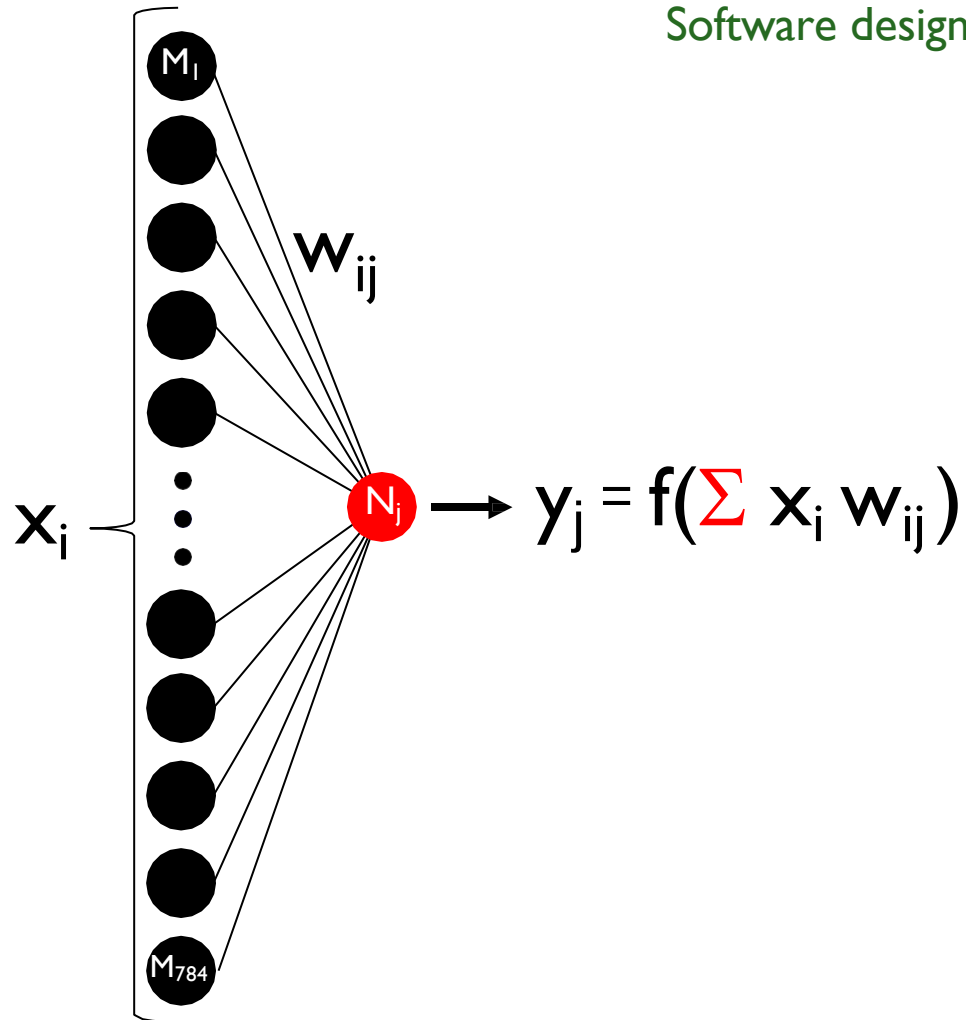
Hardware implementation



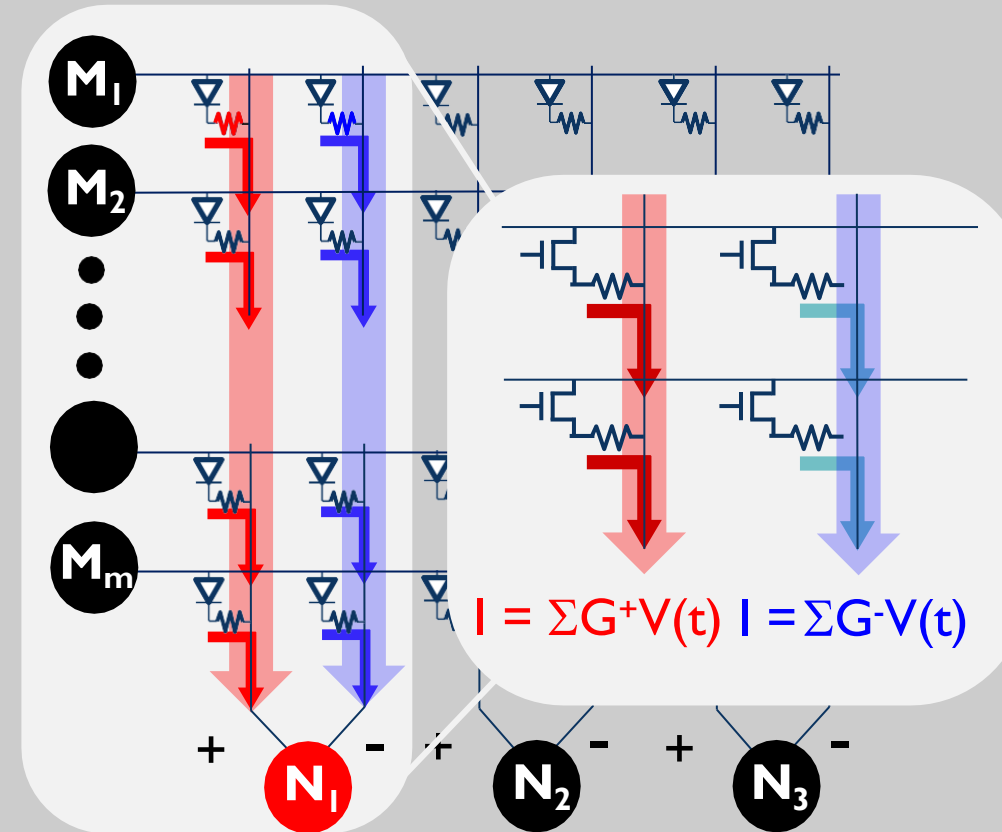
- The product $x_i w_{ij}$ is obtained using Ohm's Law

Multiply-Accumulate with NVM

Software design



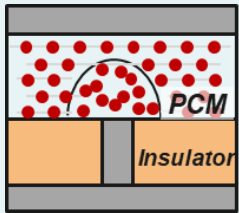
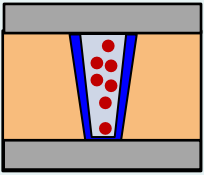
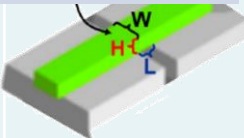
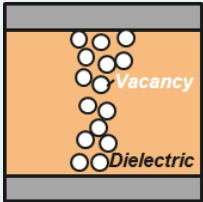
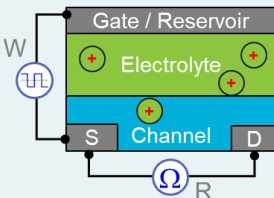
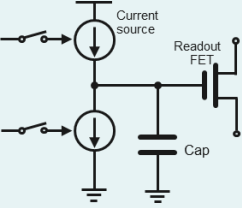
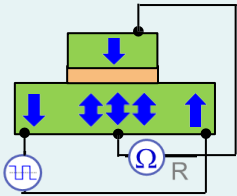
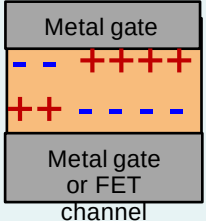
Hardware implementation



- The **sum** performed using Kirchhoff's Law

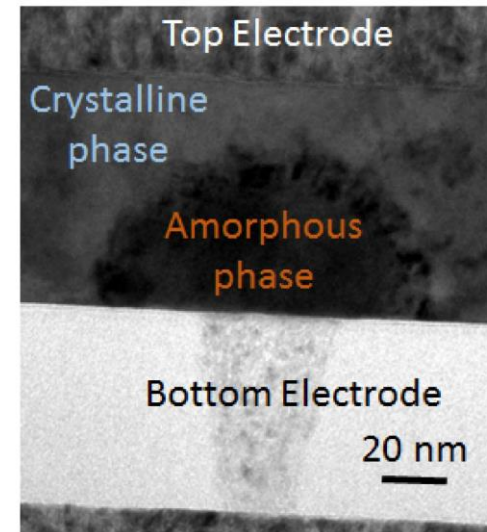
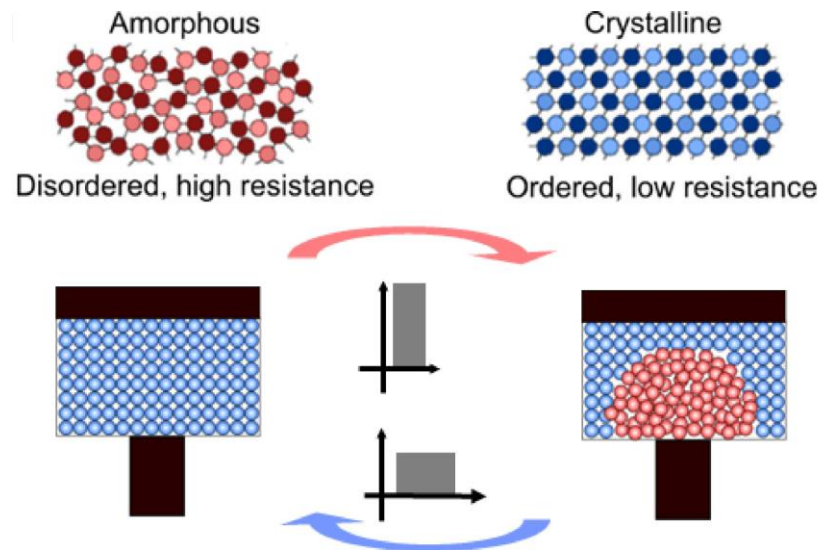
Analog Memory Options (Examples)

- No ideal device for analog computing, but many promising candidates.

Phase change memory (PCM) (crystalline/amorphous transition)			RRAM (filamentary)	ECRAM (Electrochem)	CMOS (capacitor)	MRAM & Ferroelectric (domain wall motion)	Photonics (transmission)
 mushroom	 confined surfactant	 Phase-change "bridge"	 Vacancy Dielectric	 Gate / Reservoir Electrolyte Channel S D R	 Current source Readout FET Cap	 magnetic	 Metal gate Metal gate or FET channel FE Ring resonators or Mach-Zehnder interferometers Photo-detector
+ Mature among storage-class memory			+ Bi-directional + Small	+ Symmetric + Low power	+Symmetric + Mature	+ Symmetric + Low power	+ Low power + Mature
- Abrupt reset - Asymmetry (1-cell) - Drift			- Stochastic -Asymmetry	- Open circuit voltage - Ion motion	- Cell size - Retention	- Domain wall control	- Cell size - Scalability (# of neurons)

Why Phase Change Memory (PCM)?

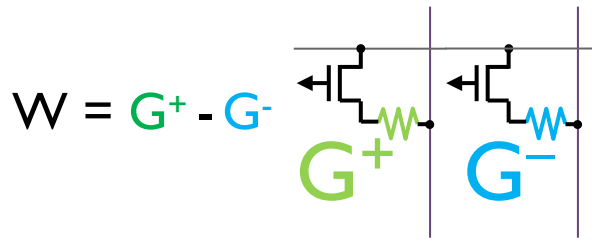
- Mature memory technology (large-scale demos & products)
- Large resistance contrast (allows more analog states)
- Much longer endurance than Flash
- Good physical understanding of device non-idealities, such as conductance drift



Outline

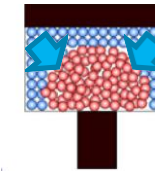
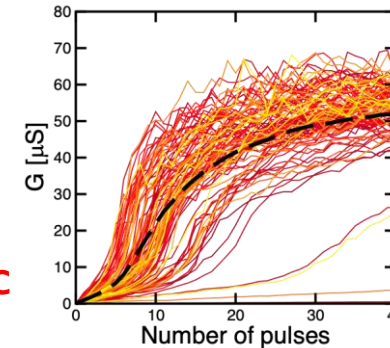
- Introduction – Deep Neural Networks (DNN) and Analog Memory
- Phase Change Memory for DNN Training and Inference
- Energy Efficiency for Analog Memory-Based Techniques
- Summary

For Training: Achieving Software Accuracy



Problem: Conductance changes in PCM are ...

- Uni-directional
- Stochastic
- Non-linear → asymmetric



What do we really want?

For training

- Gentle, symmetric conductance changes

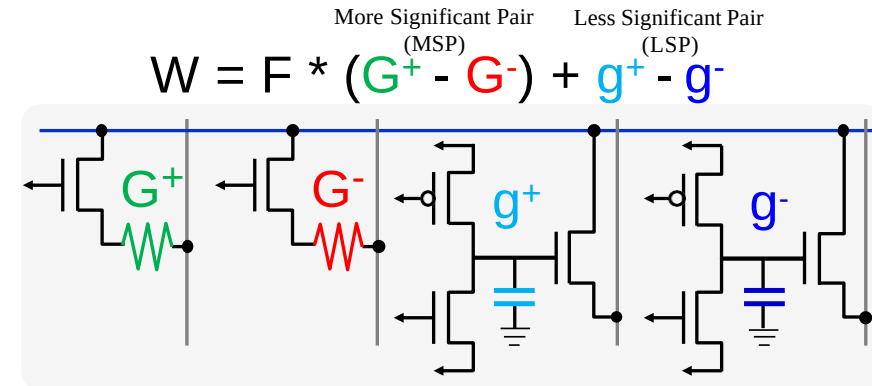
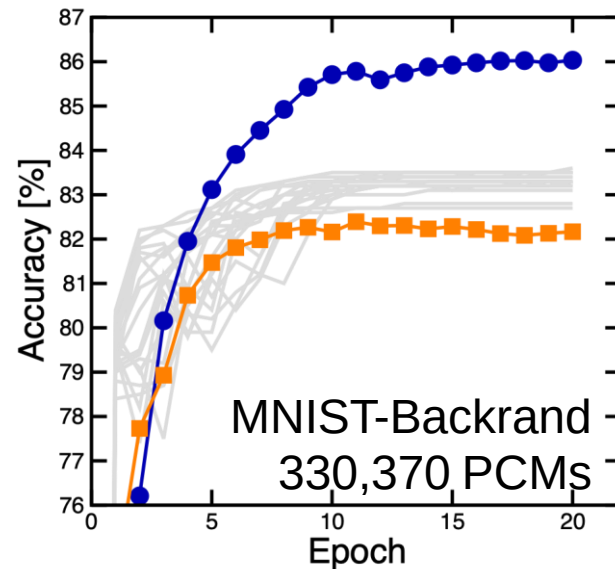
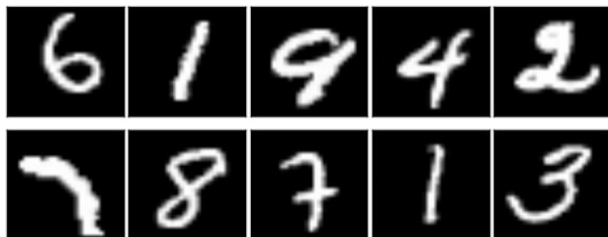
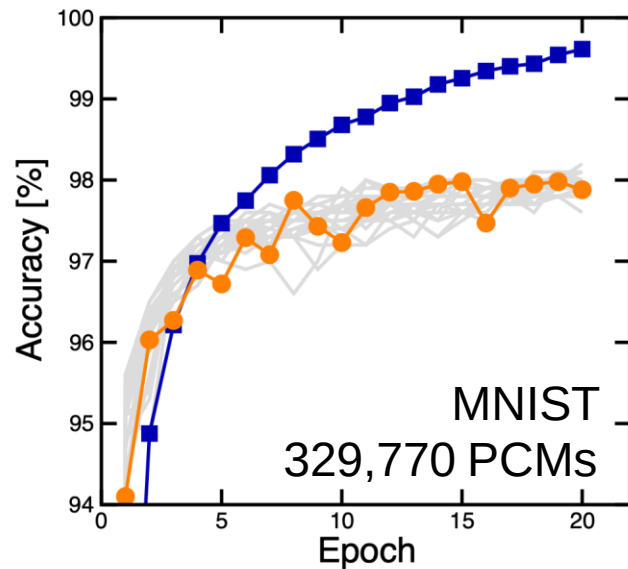
Our published results in DNN training w/ PCM on MNIST dataset

2014 – IEDM → **82%** w/ “mixed-hardware-software” experiment

2018 – *Nature* → **98%** (e.g., software-equivalent) w/ new unit-cell

Accuracy on MNIST Datasets

- Software-equivalent training accuracy achieved with 2T2R+3T1C unit cell and “polarity inversion” technique

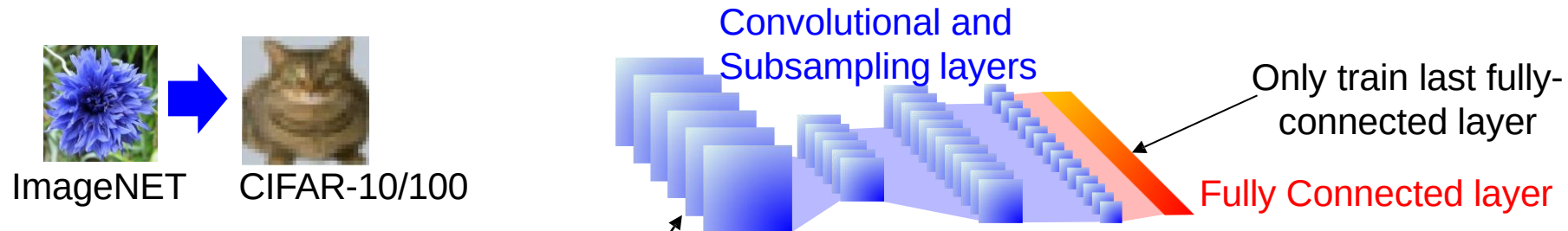
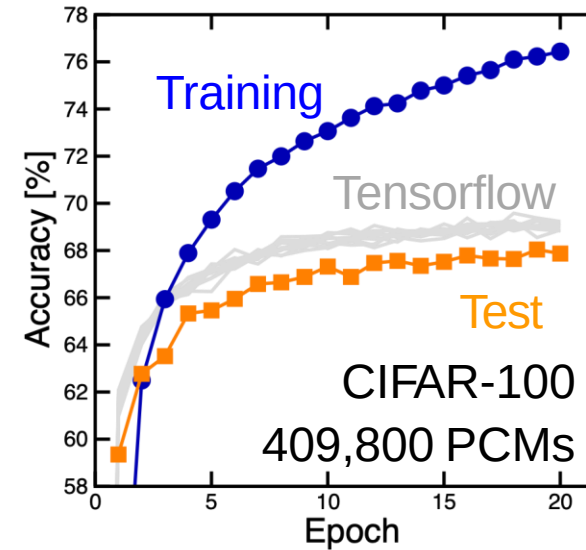
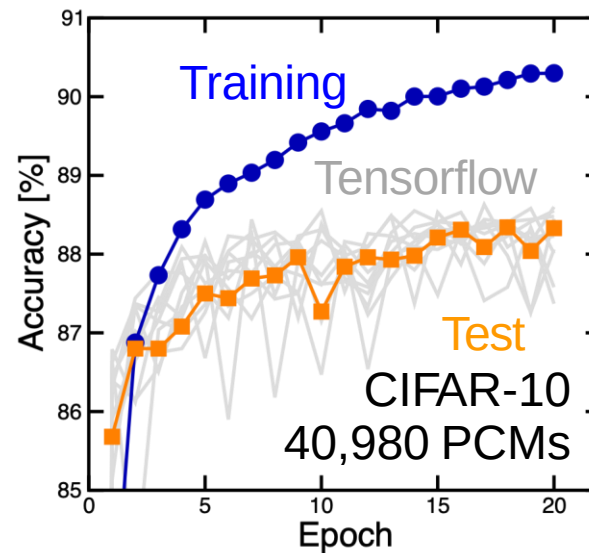


- Symmetry** → Weight update performed on g^+ – g^- shared among many columns
- Dynamic Range** → Gain factor F (e.g. $F = 3$)
- Non-Volatility** → Weight transferred to PCMs infrequently (every 1000s of images)
- “CMOS variabilities”** → Counteracted by “Polarity Inversion” technique

S.Ambrogio et al., *Nature*, 558, 60 (2018)

Transfer learning

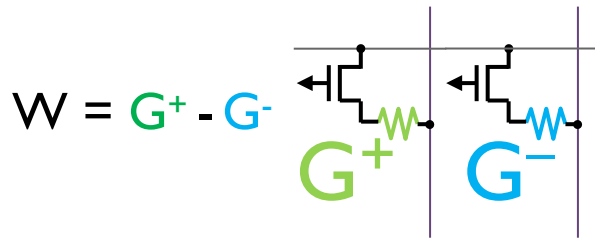
ImageNet to CIFAR-10/100



Transfer Learning: Use pre-trained, scaled weights from ImageNET for convolution layers

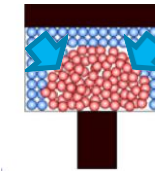
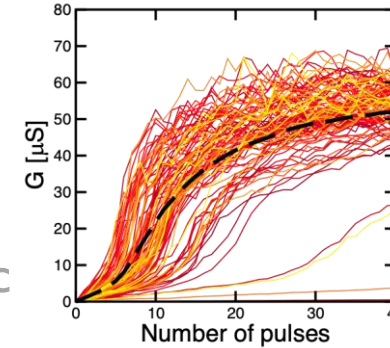
S.Ambrogio et al., *Nature*, 558, 60 (2018)

For Inference: Addressing PCM Non-Idealities



Problem: Conductance changes in PCM are ...

- Uni-directional
- Stochastic
- Non-linear → asymmetric



What do we really want?

For training

- Gentle, symmetric conductance changes

For inference...

- Precise tuning
- High yield
- No change over time

Our recent results in DNN inference w/ PCM

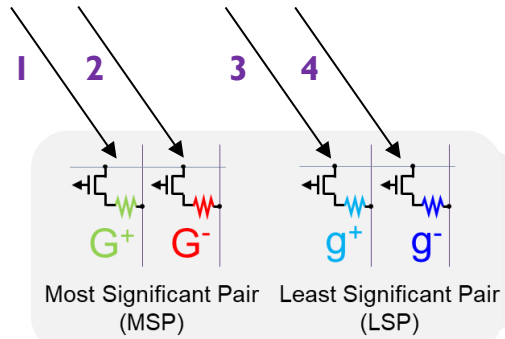
Jan 2019 – *Adv. Electr. Mater.* → programming schemes for 4 PCM devices

June 2019 – *VLSI Tech. Symp.* → software-equivalence in “mixed-hardware-software” experiment for Long-Short Term Memory (LSTM)

Dec 2019 – *IEDM* → effects of PCM “resistance drift” on DNN accuracy

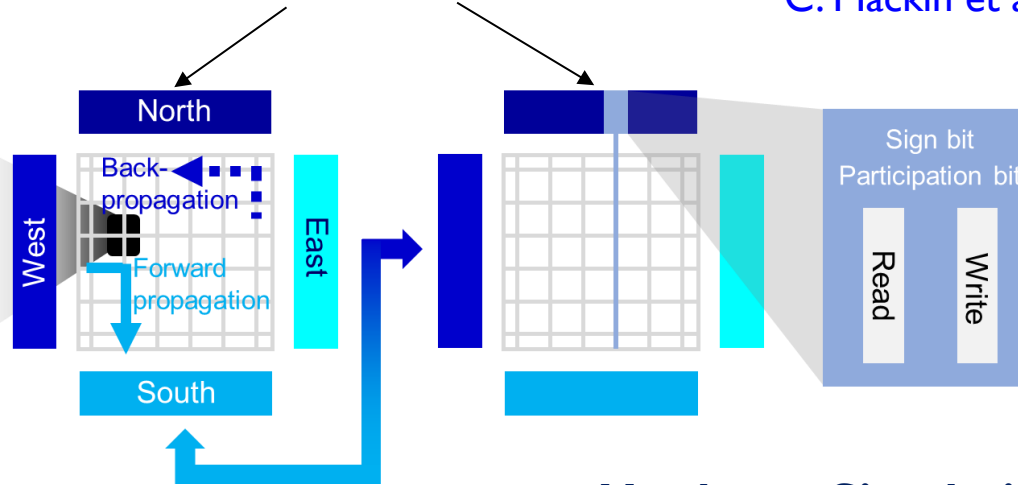
Programming of Multi-PCM Weights

Four Phases



$$W = F \times (G^+ - G^-) + g^+ - g^-$$

Physical Location

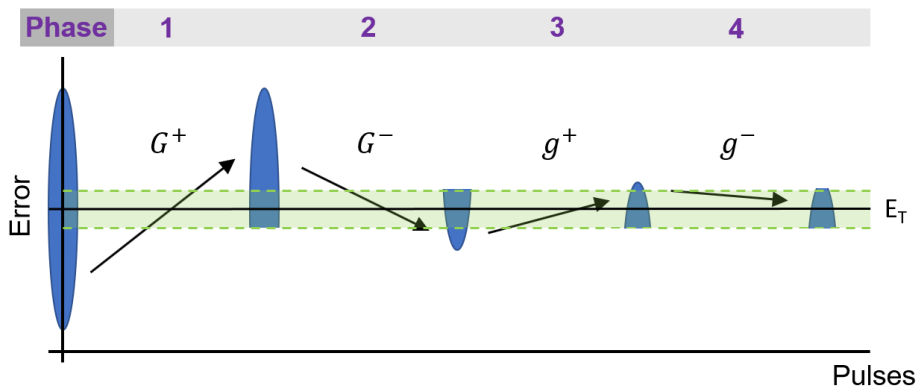


C. Mackin et al., *Adv. Electr. Mater.*, 1900026 (2019)

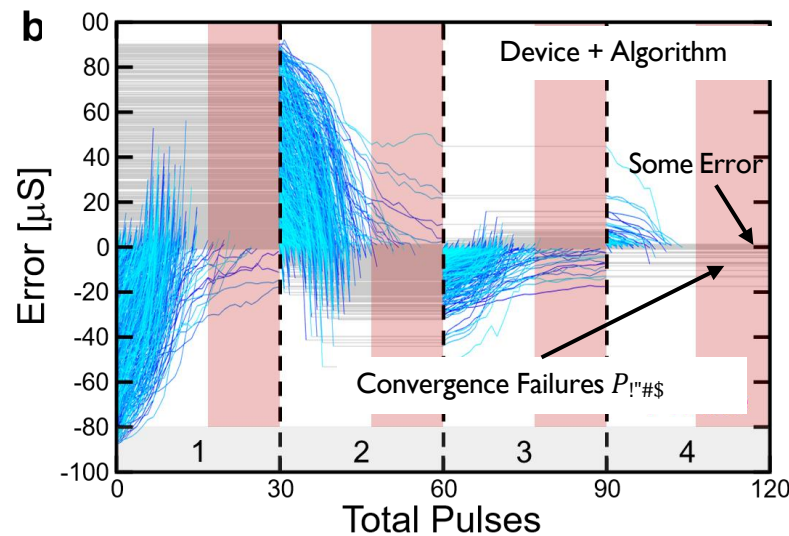
- Minimize computation expense
- Minimize area cost
- Program entire row in parallel
- 2 bits per weights (p,s)

$$Error = W - W_i$$

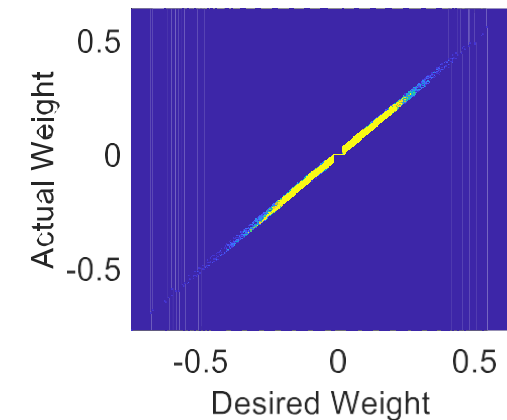
$$Error \rightarrow 0$$



Hardware Simulation



Correlation



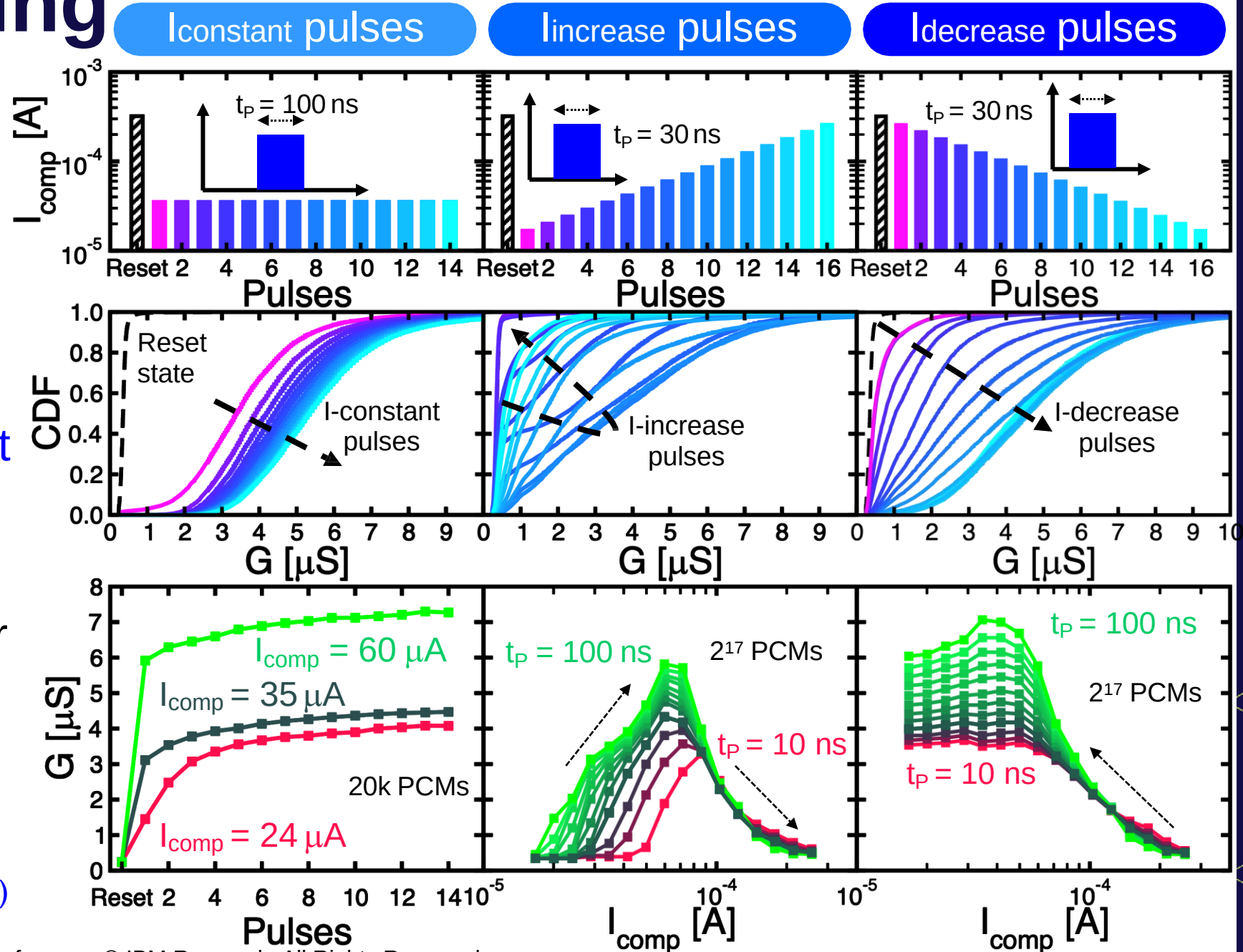
Programming PCM

Three programming schemes

- Constant Current
- Increase Current
- Decrease Current

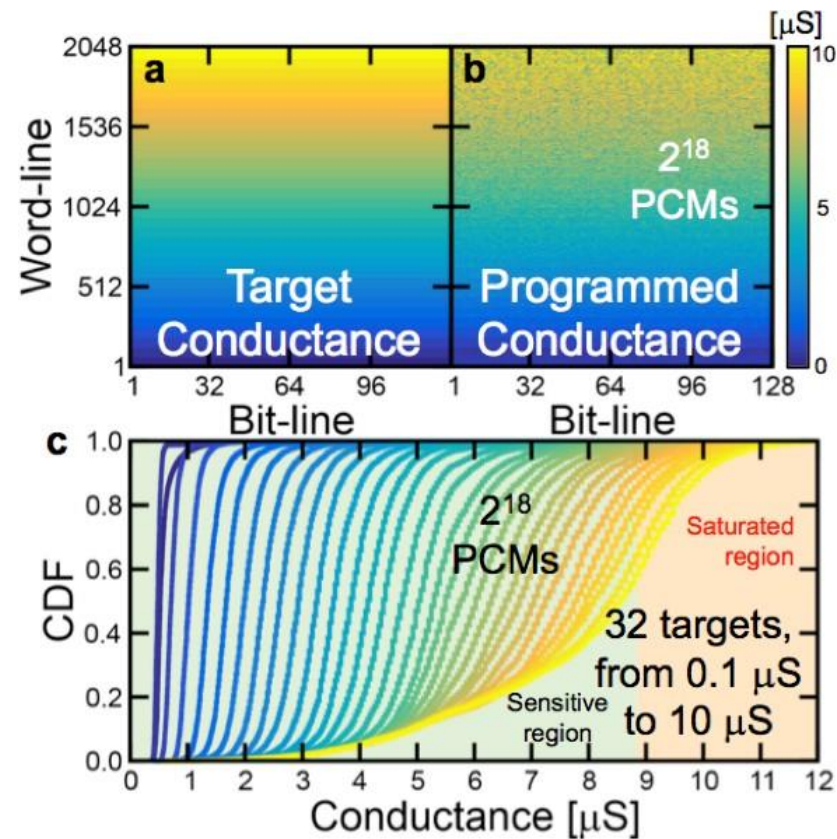
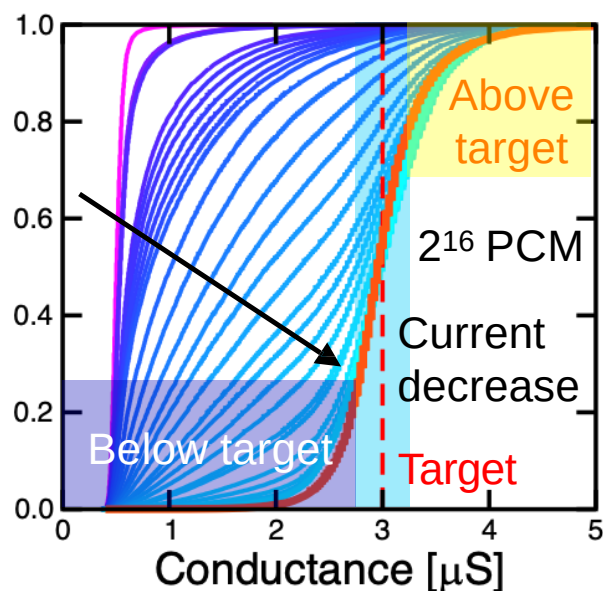
Decrease Current Pulses provide faster and monotonic convergence to analog G states

S.Ambrogio, IEDM, 6.1 (2019)



Closed Loop Parallel Programming with Device Variation

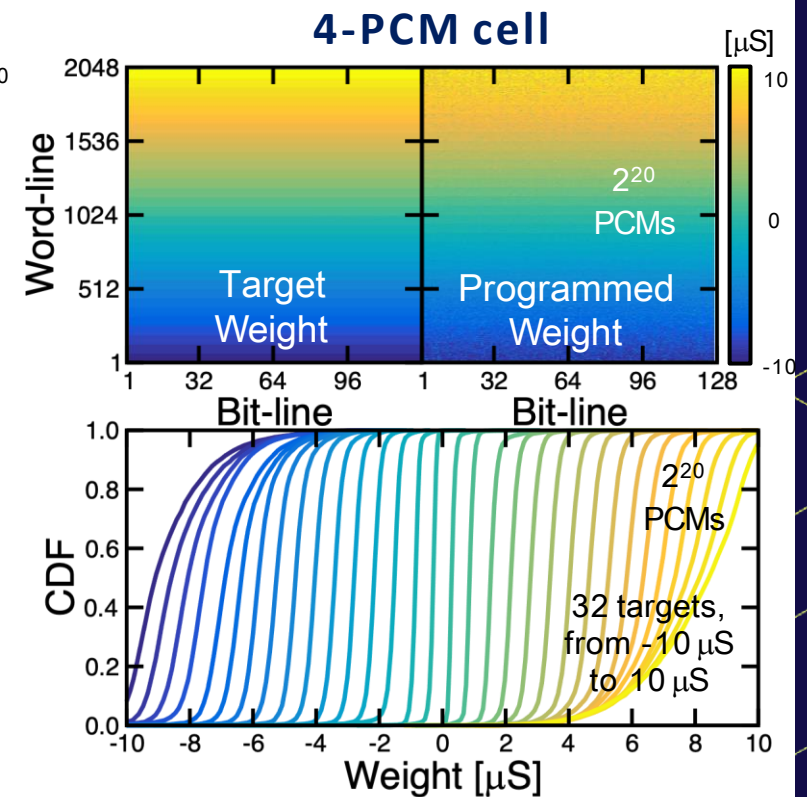
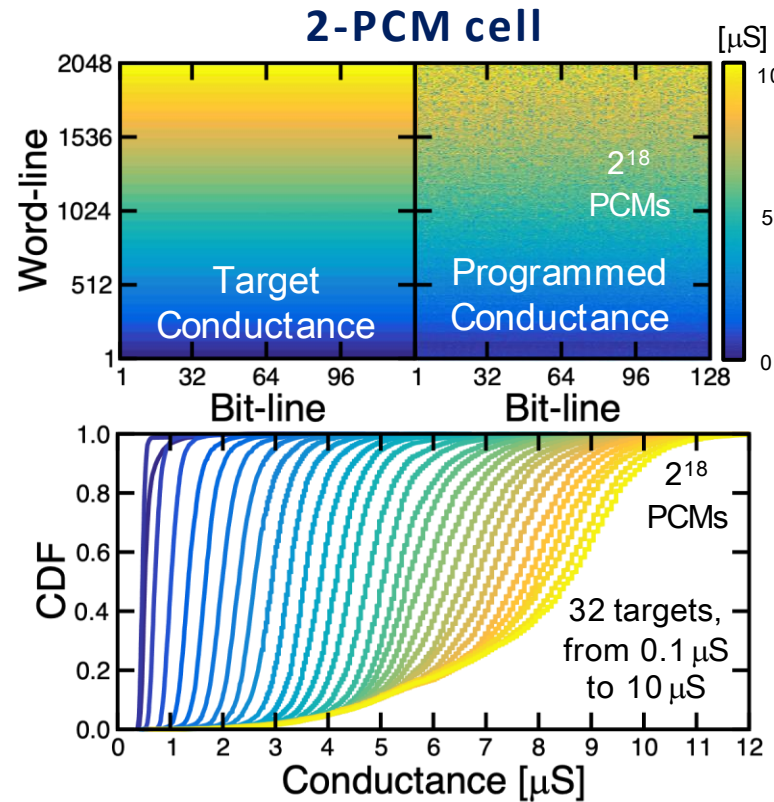
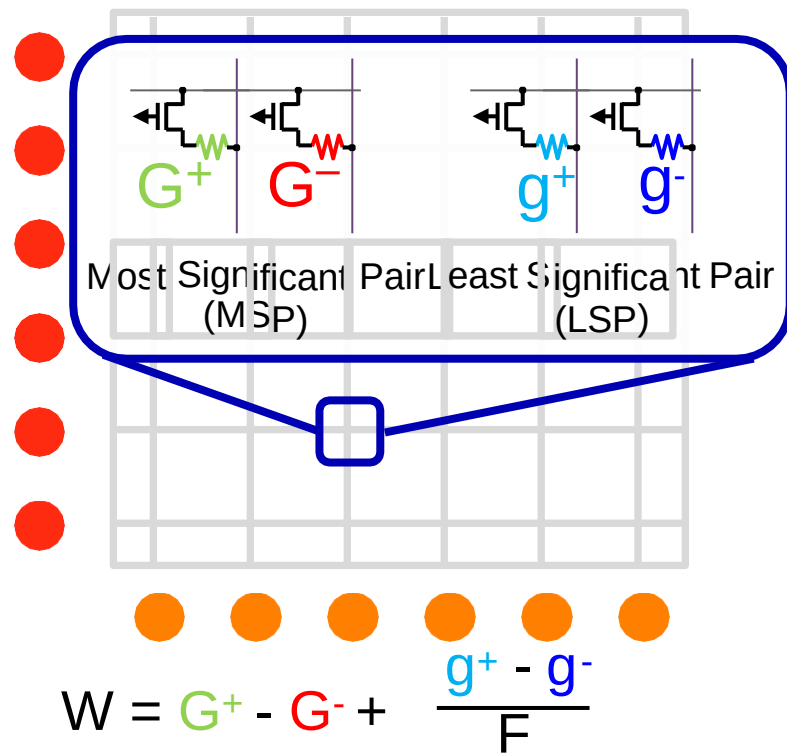
- An iterative RESET programming scheme to program phase change memory (PCM) is more tolerant to device-to-device variations



H. Tsai, VLSI Symposium, T82-83 (2019)

Mapping of Multi-PCM Weights

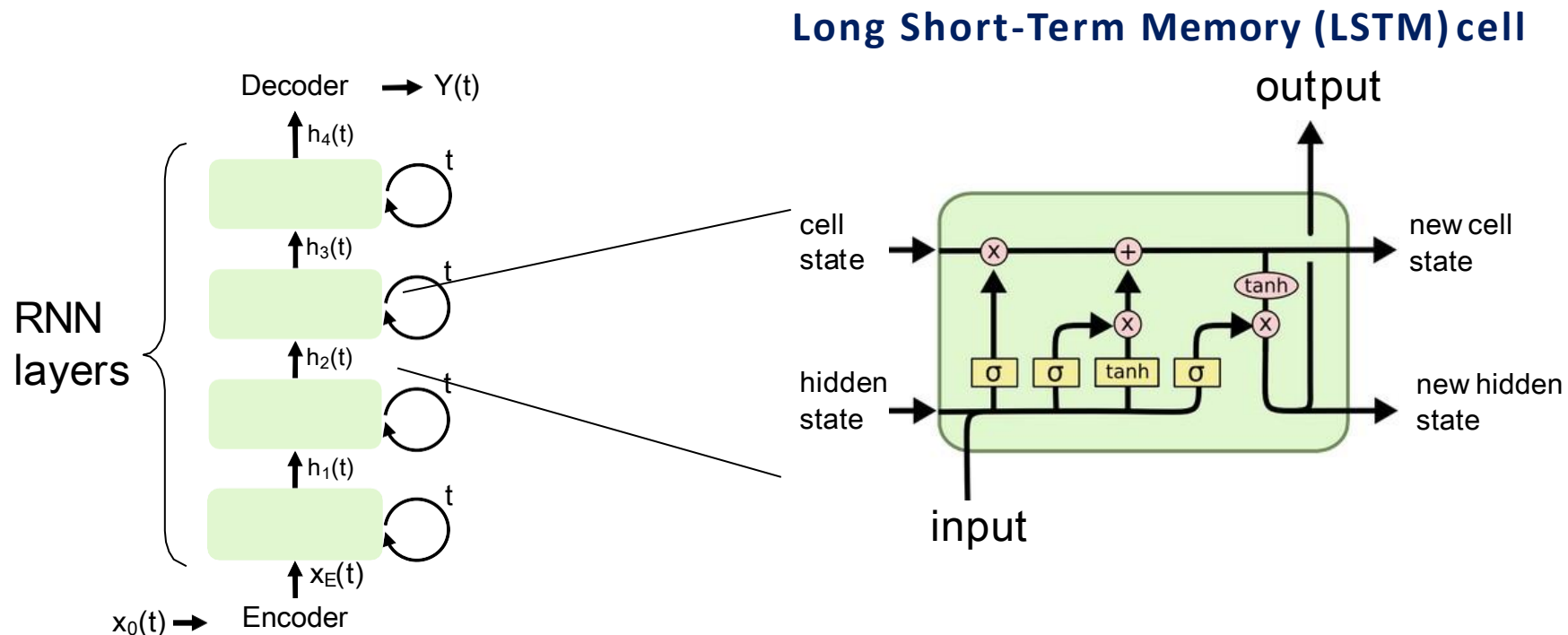
- Mapping weights to 4 phase change memory (PCM) devices improves resilience to **write noise** and **conductance saturation**



H. Tsai, VLSI Symposium, T82-83 (2019)

Long-Short Term Memory (LSTM)

- Long-Short Term Memory (LSTM) networks are use extensively for sequence modeling, e.g., speech recognition and translation
- LSTMs consist of mostly fully connected networks that are well suited for analog acceleration



Language Modeling with LSTM

- Task: Predict the probability of the next character or word
- Training is supervised, but no labeling is needed
- Performance is measured by cross-entropy loss or perplexity

Word Based Prediction

$x(t)$ = And so it was indeed: she was now only ten inches high, [...]

And so it was indeed ...

LSTM "ideal" prediction

so it was indeed she ...

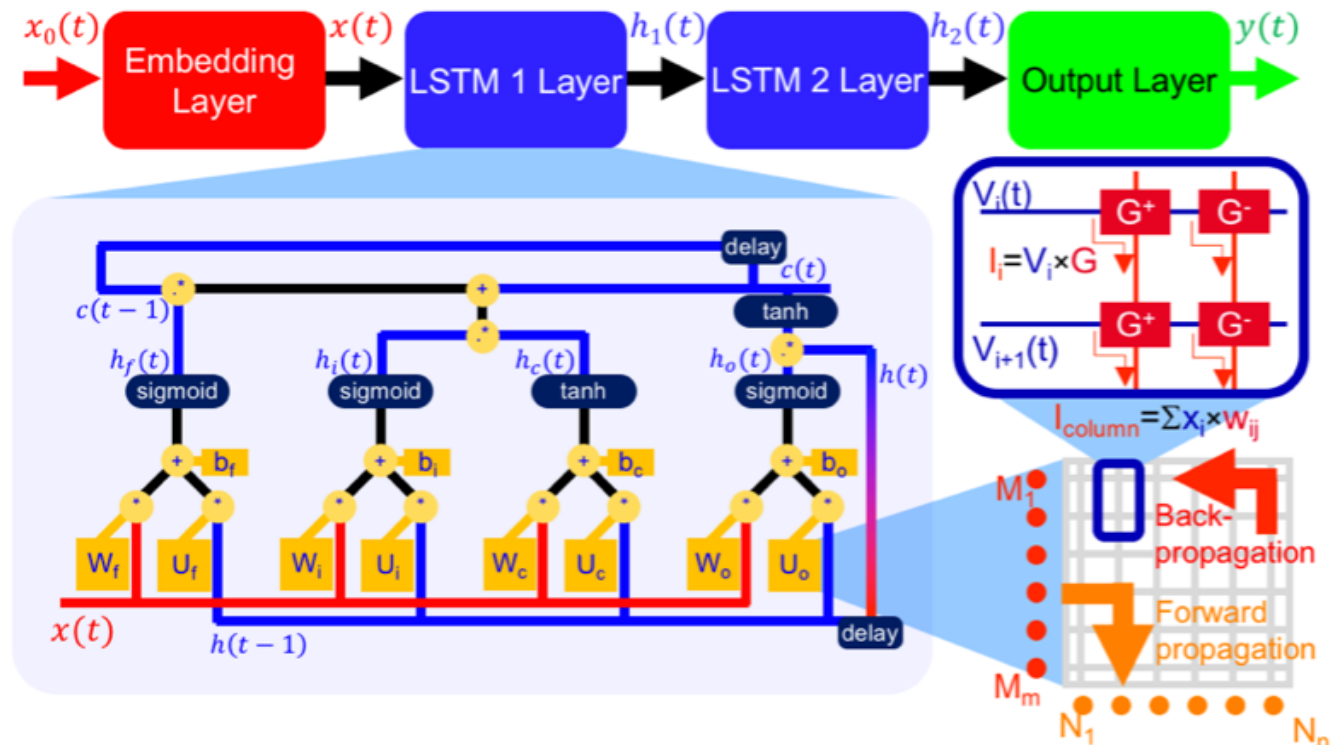
Discretized Time

Output $y(t) = W_y * h(t) + b_y$

Probability $probability(t) = softmax[y(t)]$

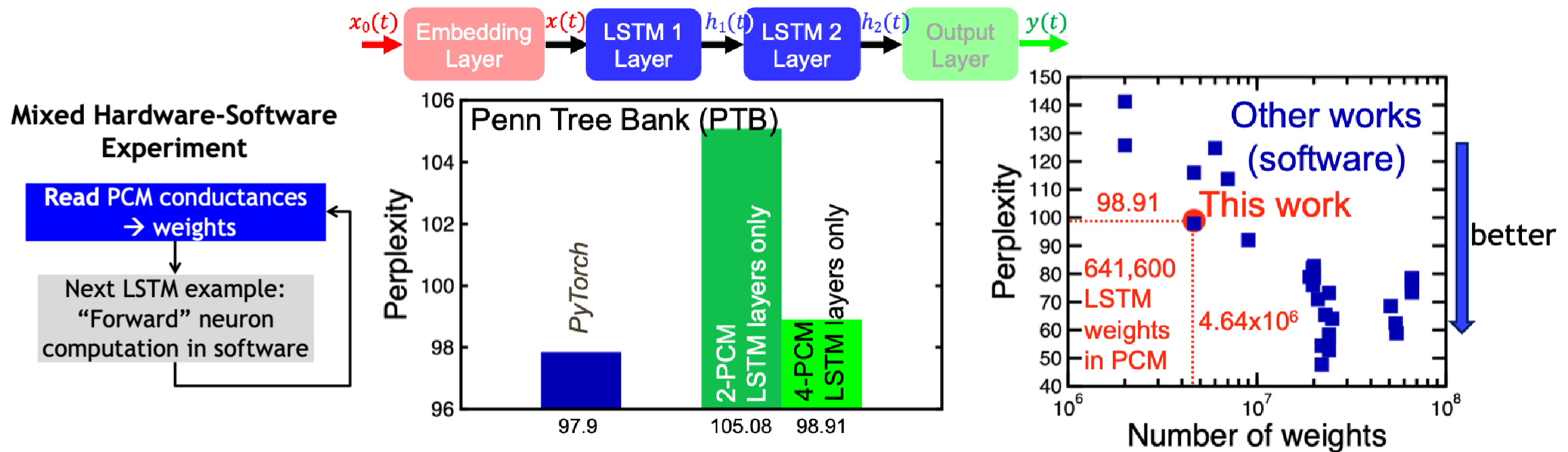
Cross-Entropy Loss =
 $-\log(\text{probability for correct answer})$

Perplexity = $\exp(\text{Cross-Entropy Loss}) =$
 $1/\text{probability for correct answer}$



Mixed Hardware-Software Experiments with Long-Short-Term Memory (LSTM)

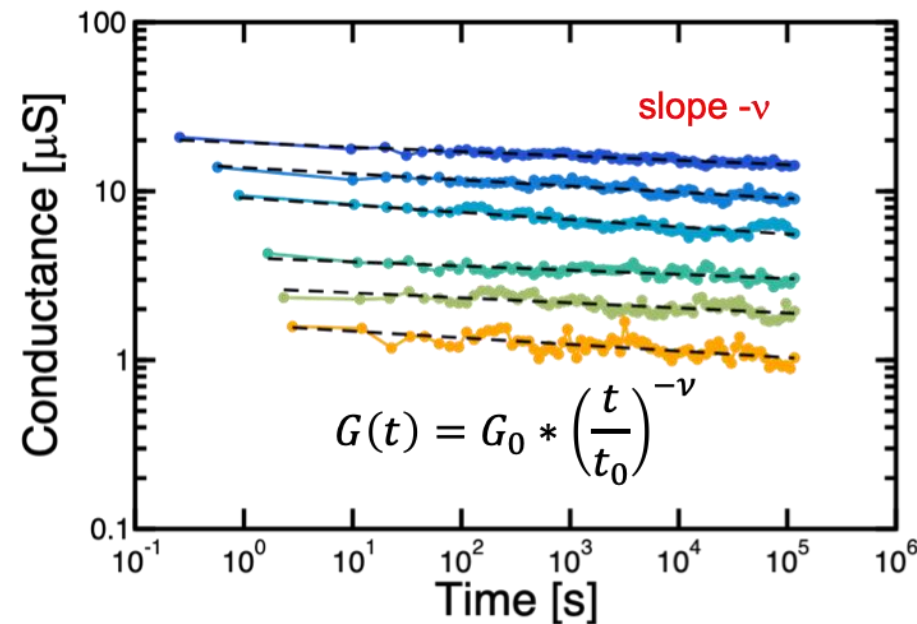
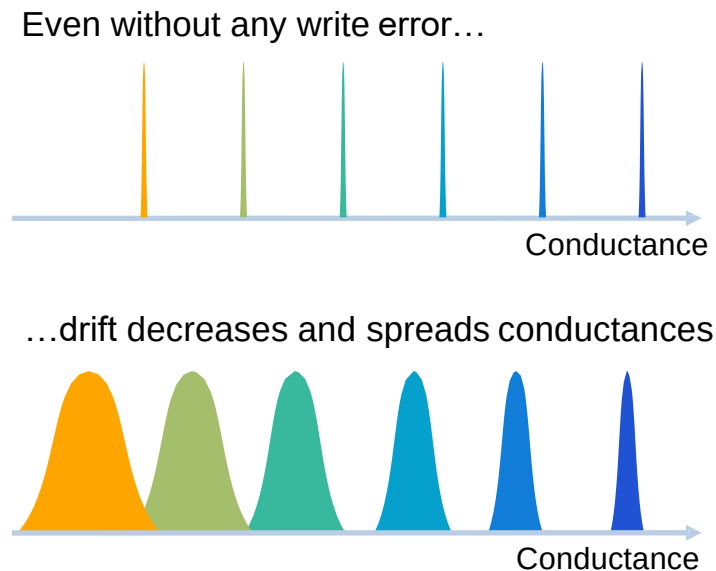
- Software-equivalent accuracy was achieved on commonly used language modeling benchmarks, with 2.5M PCM devices in weights



H. Tsai, VLSI Symposium, T82-83 (2019)

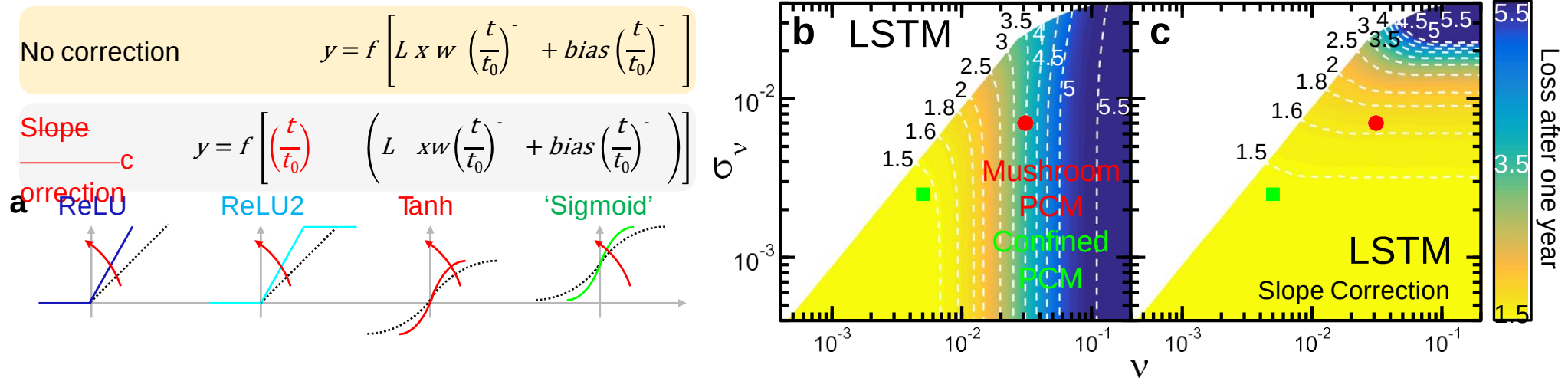
Conductance Drift

- As the amorphous state relaxes, PCM conductance gradually decreases
- PCM drift can be quantified with an exponential time dependence with a drift coefficient ν



Drift Impact and Slope Correction

- PCM drift causes weight decrease at different rates from device to device, which increases LSTM loss over time
- Slope correction: tuning the activation function leads to signal restoration

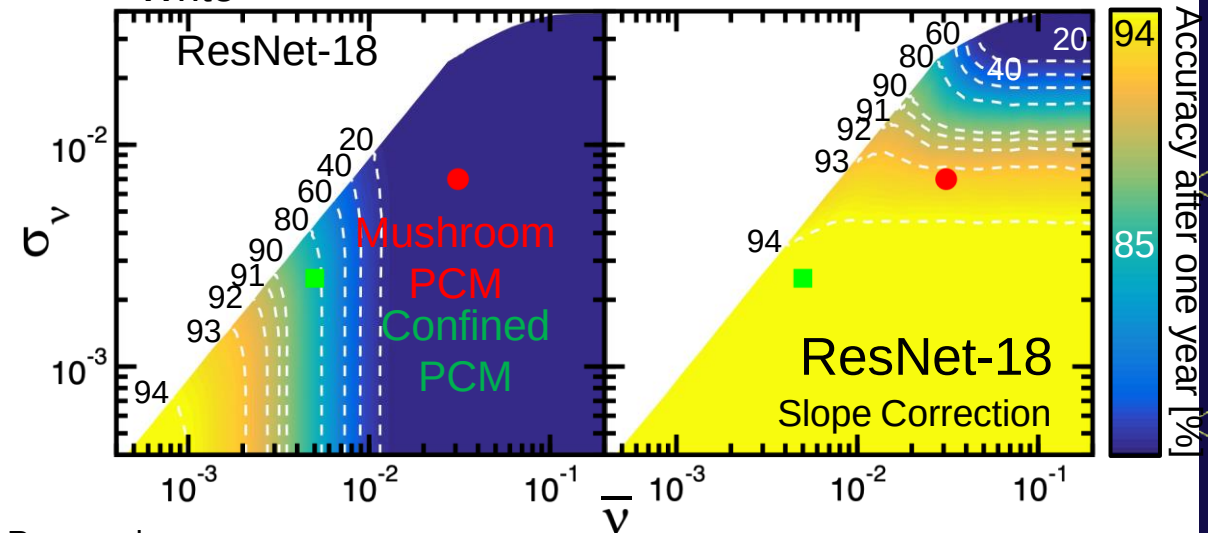
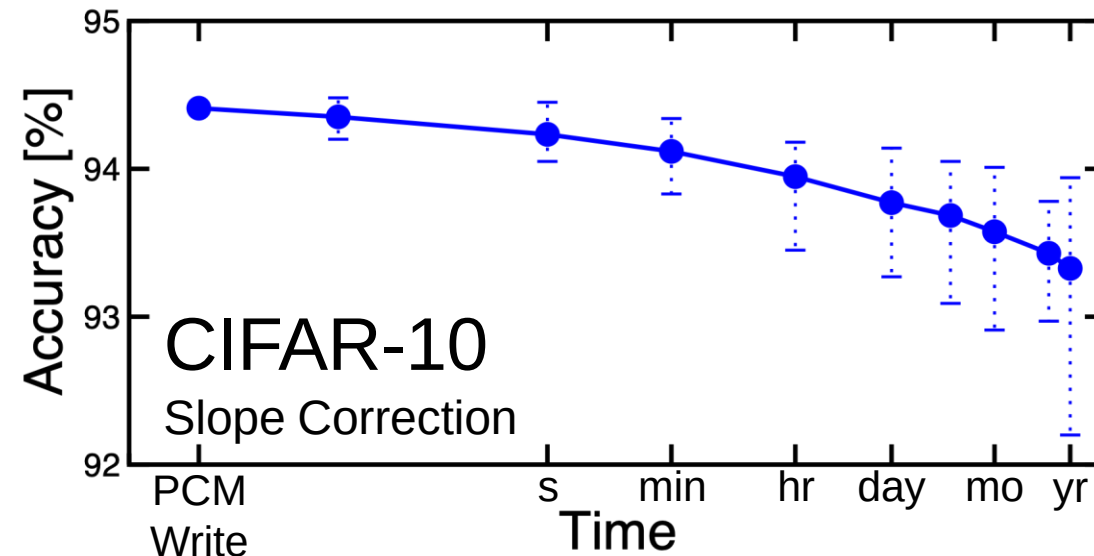
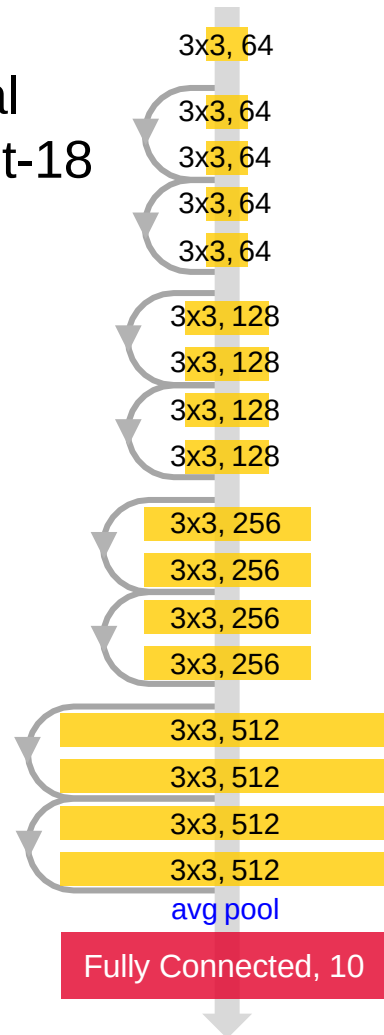


S.Ambrogio, IEDM, 6.1 (2019)

Impact of PCM Drift on ResNet-18

- Impact of drift is much stronger since every weight is reused many times

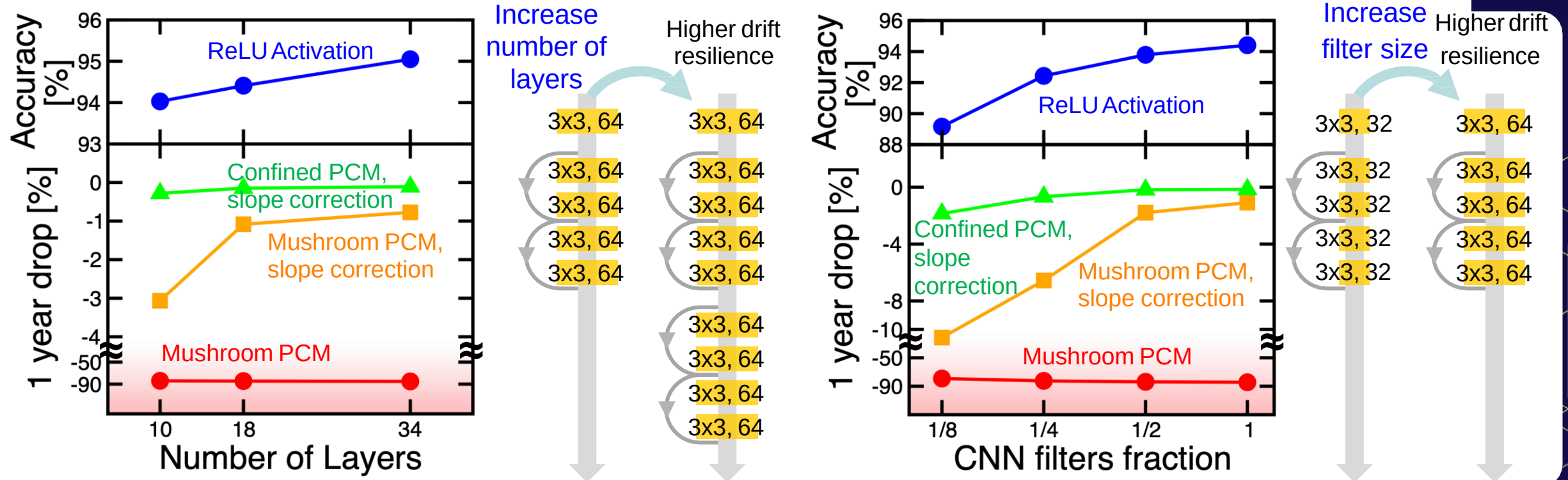
Convolutional Neural
Network (CNN): ResNet-18
Image: CIFAR-10



Dependence on Network Design

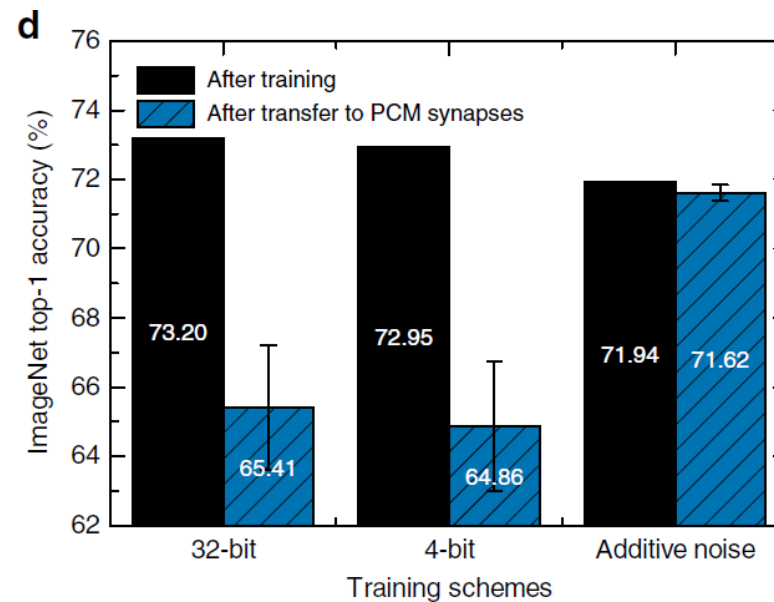
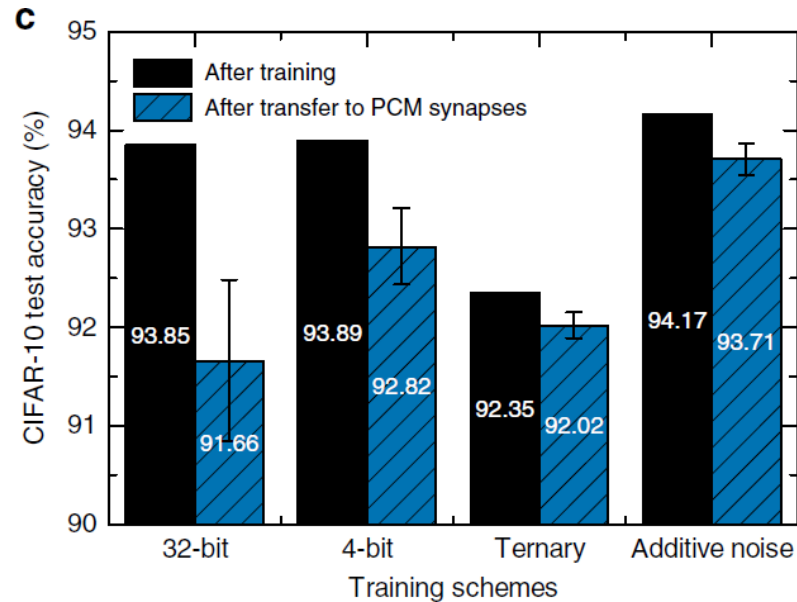
- Increasing number of hidden layers or size of hidden layer leads to increased drift resilience – IF slope correction is used

ResNet-18 Conv Network for CIFAR10



Noise-aware DNN training for Analog HW

- ResNet-34 trained on CIFAR-10 and ImageNet datasets
- Additive noise re-training can improve robustness of model and recover loss in inference accuracy



V. Joshi, Nature Communications (2020)

Outline

- Introduction – Deep Neural Networks (DNN) and Analog Memory
- Phase Change Memory for DNN Training and Inference
- Energy Efficiency for Analog Memory-Based Techniques
- Summary

Hardware Approach for Energy Efficiency

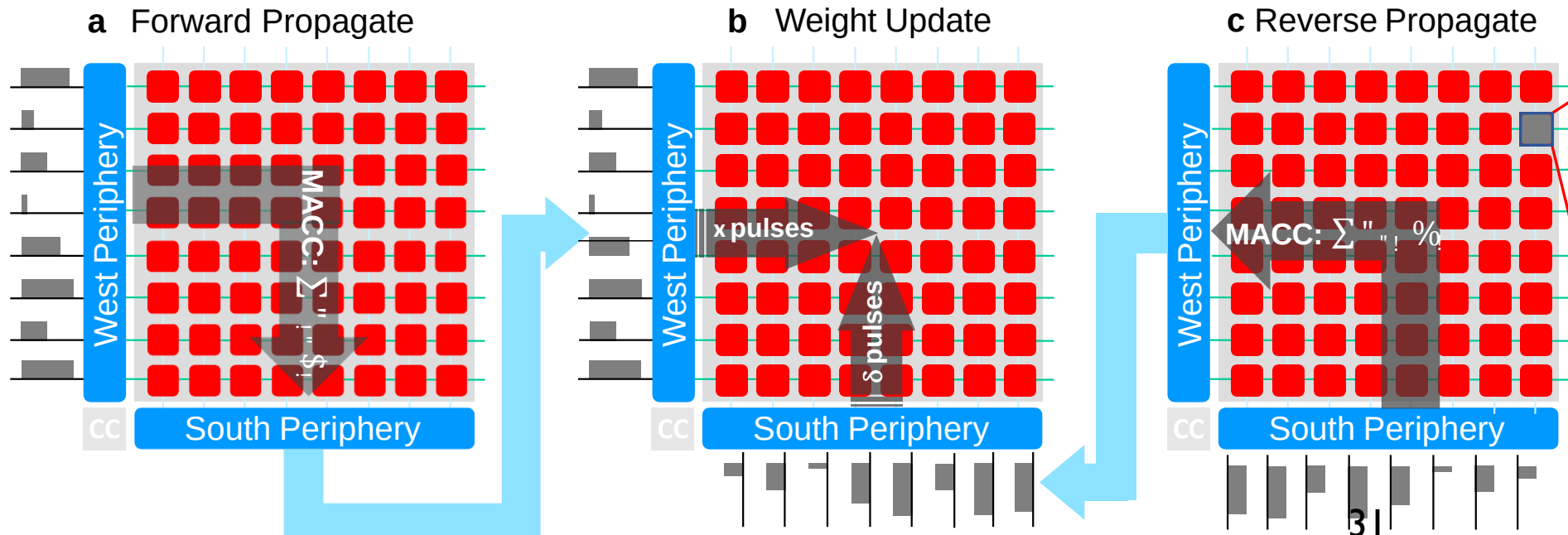
- 1) Parallelism is key
- 2) Avoiding ADC (Analog-to-Digital Conversion) saves time, power & area
- 3) Do the necessary computations (squashing functions) but be as "approximate" as you can
- 4) Develop efficient and reconfigurable routing strategies to get vectors of data from the bottom of one array to the edge of the next one

AI hardware acceleration
with analog memory:
micro-architectures for low
energy at high speed

This paper presents innovative micro-architectural designs for multi-layer Deep Neural Networks (DNNs) implemented in crossbar arrays of analog memories. Data are transferred in a

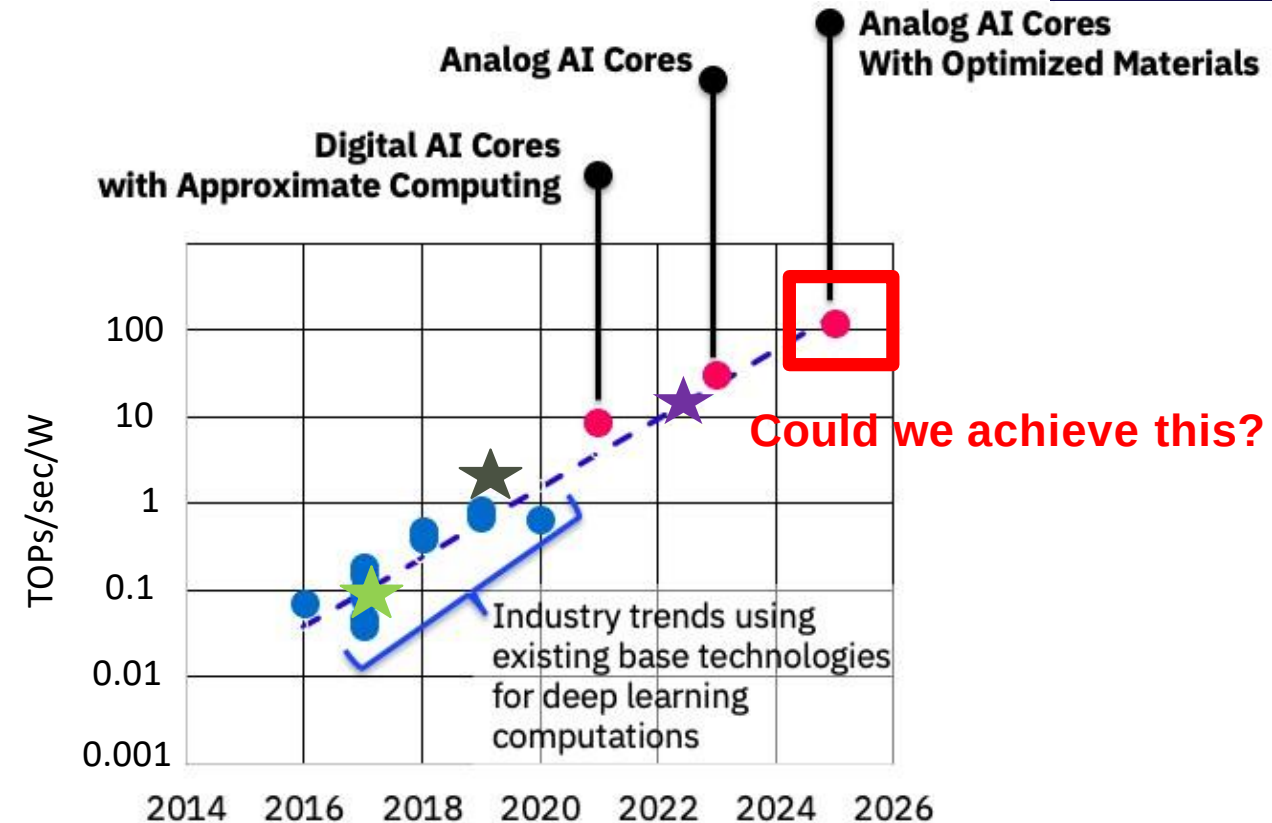
IBM J. R&D, IEEE vol. 63, pp. 8:1-8:14, 1 Nov.-Dec. 2019.

Hung-Yang Chang
Prithish Narayanan
Scott C. Lewis
Nathan C. P. Farinha
Kohji Hosokawa
Charles Mackin
Hsinyu Tsai
Stefano Ambrogio
An Chen
Geoffrey W. Burr



Where are we on the Roadmap?

- **NVIDIA V100:** 0.1 TOPs/sec/W
- **Google TPU1:** 2.3 TOPs/sec/W
 - Inference Only
 - NOT including data movement
- **IBM internal Analog designs:**
 - MNIST : **15.2** TOPs/sec/W
 - PTB LSTM : **14** TOPs/sec/W



AI roadmap from IBM AI Hardware Center announcement
www.ibm.com/blogs/research/2019/02/ai-hardware-center/

How to Improve Energy Efficiency?

- 1) **Reduce average NVM conductance** → reduces array currents during Multiply-Accumulates
→ Current focus of various material and device design efforts

- 2) **Reduce technology node**

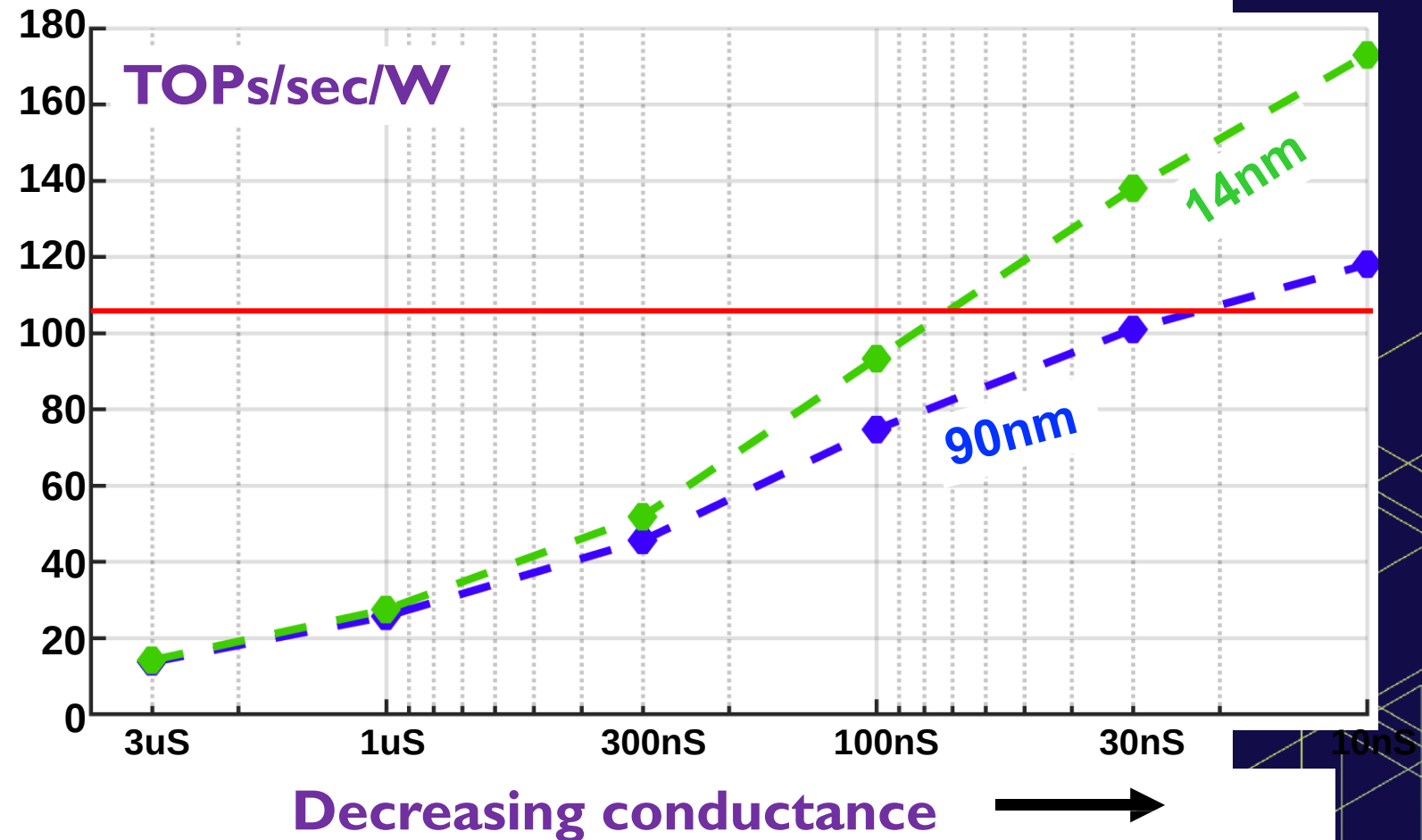
90nm → **14nm**

Benefits just from scaling
routing energy

Area efficiency for inference:
10—70 TOPs/sec/mm²

(vs. ~0.3 TOPs/sec/mm² for TPU v1: In-
Datacenter Performance Analysis of a Tensor
Processing Unit)

IBM J. R&D,
IEEE vol. 63, pp. 8:1-8:14, 1 Nov.-Dec. 2019.



Conclusion

- NVM-based crossbar arrays **can accelerate** Deep Machine Learning compared to GPUs
 - Multiply-accumulate performed at the data → saves power and time
 - But conventional NVM devices (like PCM) **are imperfect...**
- Recent training results
 - Mixed-hardware-software experiments → **software-equivalent training accuracy** (S. Ambrogio et al, *Nature*, 558, 60 (2018))
- Recent inference results
 - Programming strategies for 4-PCM-based weights (C. Mackin et al., *Adv. Electr. Mater.*, 1900026 (2019))
 - Mixed-hardware software experiments on LSTM (H. Tsai et al., *VLSI Tech. Symp.* (2019))
 - Impact of resistance drift in PCM (S. Ambrogio et al., *IEDM*, 6.1 (2019))
- Power projections based on real circuit designs
 - **100x better energy efficiency** (+ 100x speedup) on fully-connected layers (for LSTM and other networks) (H.-Y. Chang et al., *IBM J. R&D*, (2019))

htsai@us.ibm.com

Acknowledgements



Geoffrey
Burr



Pritish
Narayanan



Stefano
Ambrogio



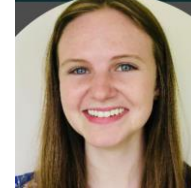
Hsinyu
Tsai



Charles
Mackin



An
Chen



Katie
Spoon



Bob
Shelby



Kohji
Hosokawa



Scott
Lewis

Management Support



Spike
Narayan



Winfried
Wilcke



Bulent
Kurdi



Jeff
Welser



Heike
Riel



Sudhir
Gowda



Dario
Gil



Wilfried
Haensch



Arvind
Kumar



Vijay
Narayanan



Jeff
Burns