Applied Machine Learning Days @ EPFL

State of the art in hardware-accelerated neural networks

Frédéric Pétrot, Lorena Anghel, Liliana Andrade
Univ. Grenoble Alpes, CNRS, Grenoble INP[*], TIMA,
F-38000 Grenoble, France

⌂ tima.imag.fr/sls/people/petrot

✉ frederic.petrot@univ-grenoble-alpes.fr

[*] Institute of Engineering Univ. Grenoble Alpes

Biology vs Electronics

The Brain: the Ultimate Autonomous System

- 1,2 to 1,4 kg, 1260 cm^3
- Consumes between 15 and 30 Watts
- 86×10^9 neurones, $\approx 10^{12}$ synapses

Biology vs Electronics

The Brain: the Ultimate Autonomous System

- 1,2 to 1,4 kg, 1260 cm^3
- Consumes between 15 and 30 Watts
- 86×10^9 neurones, $\approx 10^{12}$ synapses

J. Vitti and D. Silverman, *"Bart the Genius"*, The Simpsons, 1990

Biology vs Electronics

The Brain: the Ultimate Autonomous System

- 1,2 to 1,4 kg, 1260 cm^3
- Consumes between 15 and 30 Watts
- 86×10^9 neurones, $\approx 10^{12}$ synapses

J. Vitti and D. Silverman, *"Bart the Genius"*, The Simpsons, 1990

Human Brain Project (EU Flagship)

- 16,000 neurones per 1 Watt chip
- 5.375 MW/brain
- 1.2×10^9 € from Europe: $\approx$ 1.4 cent of €/neurone

Biology vs Electronics

The Brain: the Ultimate Autonomous System

- 1,2 to 1,4 kg, 1260 cm^3
- Consumes between 15 and 30 Watts
- 86×10^9 neurones, $\approx 10^{12}$ synapses
- (Energy comes from eggs, honey and mushrooms)

J. Vitti and D. Silverman, *"Bart the Genius"*, The Simpsons, 1990

Human Brain Project (EU Flagship)

- 16,000 neurones per 1 Watt chip
- 5.375 MW/brain
- 1.2×10^9 € from Europe: $\approx$ 1.4 cent of €/neurone

Biology vs Electronics

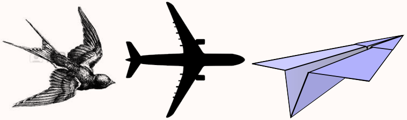
The Brain: the Ultimate Autonomous System

- 1,2 to 1,4 kg, 1260 cm^3
- Consumes between 15 and 30 Watts
- 86×10^9 neurones, $\approx 10^{12}$ synapses
- (Energy comes from eggs, honey and mushrooms)

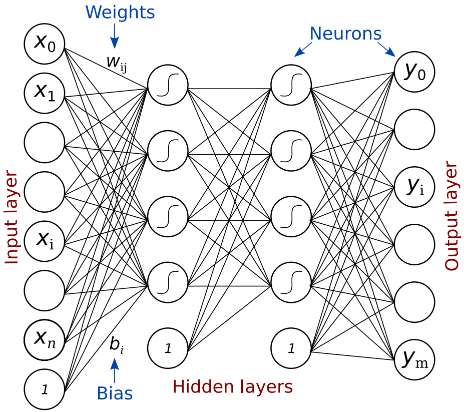
J. Vitti and D. Silverman, *"Bart the Genius"*, The Simpsons, 1990

Human Brain Project (EU Flagship)

- 16,000 neurones per 1 Watt chip
- 5.375 MW/brain
- 1.2×10^9 € from Europe: $\approx$ 1.4 cent of €/neurone



Classical Deep Neural Network (DNN) Topology



$$u_i = \sum_{i=1}^{n} x_i \, w_{ij} \qquad y_i = \mathbf{act}(u_i + b_i)$$

Two phases

- Learning, off-line, floating point
- Inference, run-time, floating point

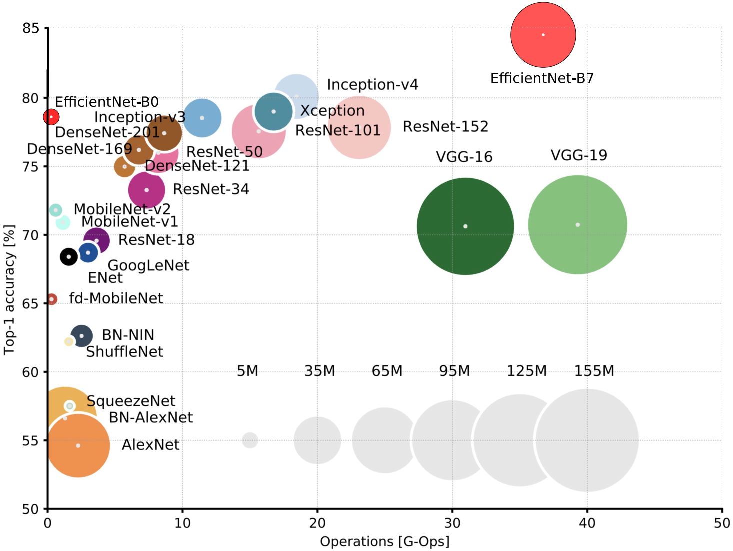
Weights

- Values “found” during learning
- Values used during inference

Issues

- Complex learning algorithms: GPU farms
- Limited computing power during inference

CNN Models: Accuracy, Operations and Weights



A. Canziani, E. Culurciello, A. Paszke, *"An Analysis of Deep Neural Network Models for Practical Applications"*, 2017 (EfficientNet-B0/B7 added by myself)

Hardware Accelerated Neural Network

Our focus? Inference!

What's the interest, by the way?

Local computation!

- Energy
  No router, cloud server, ...
  ⇒ Huge constraint in Edge Computing
  ⇒ Worse in IoT
  ⇒ Transmitting data costs energy
- Latency
  Immediate response, no dead zone, no network reliability issue, ...
- Privacy
  No storage in someone else's servers
  Neither wire nor wireless sniffing possible

Challenges in Embedded and Hardware Accelerated Neural Networks

Constraints on HW Accelerated Neural Networks

- Accuracy needs depend on the application
- Silicon resources:
  $\Rightarrow$ Computations to perform
  $\Rightarrow$ Weights storage and access
- Energy efficiency
  Typical constraints :
  - 10-100 uW for wearables,
  - 10-100 mW for phones,
  - 1-10 W for plugged devices

Challenges in Embedded and Hardware Accelerated Neural Networks

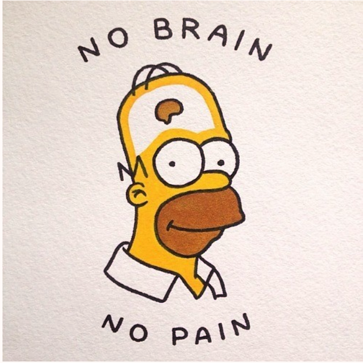
Constraints on HW Accelerated Neural Networks

- Accuracy needs depend on the application
- Silicon resources:

  ⇒ Computations to perform

  ⇒ Weights storage and access
- Energy efficiency

  Typical constraints :
  - 10-100 uW for wearables,
  - 10-100 mW for phones,
  - 1-10 W for plugged devices



Computation Demanding

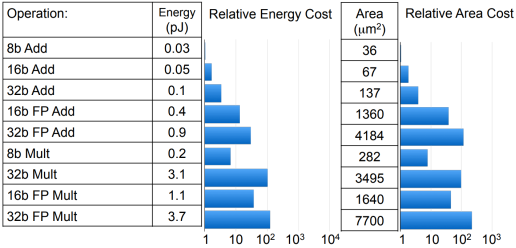
Inference involves a lot of computation...

- Elevated number of floating point (FP) operations

  $0.5G \leq$ Nb of FLOPs $\leq 40G$

- Floating point operations are energy and area costly

(My 4 core-i7 PC ~120 GFLOPs ⇒ 30 GFLOPs/core.)

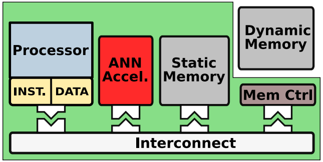


Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700

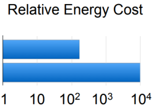
"*Hardware Architectures for Deep Neural Networks*", ISCA Tutorial, 2017

Memory demanding

Inference involves a lot of memory access...



Operation:	Energy (pJ)
32b SRAM Read (8KB)	5
32b DRAM Read	640



"Hardware Architectures for DNN", ISCA Tutorial, 2017

- Memory stores millions of (64-bit) weights
  ⇒ 4M (GoogLeNet), 60M (AlexNet), 130M (VGG)
- Memory access becomes the bottleneck
  ⇒ Each op needs 2 operands and produces a result
- An “elevated” power consumption is involved

Coping with GFLOPs and GBytes

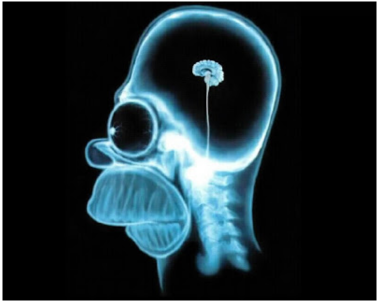
Alternatives: trade FLOPs for (some) accuracy loss

Simplify the operations

- Avoid sigmoid, batch normalization and stuff
- FP arithmetic is not HW friendly
  $\Rightarrow$ Use data-types that are not 64-bit floats

Alternatives: trade bytes for (some) accuracy loss

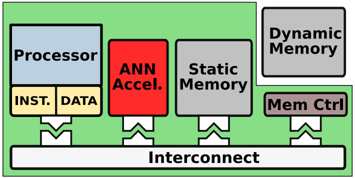
- Use “small” data types
- Integrate many memory cuts with processing elements and use them wisely
- Integrate computation into the memory itself



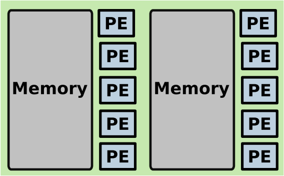
K. Usher, *"The Dwarf in the Dirt"*, Bones, 2009

(Energy comes from donuts + beer)

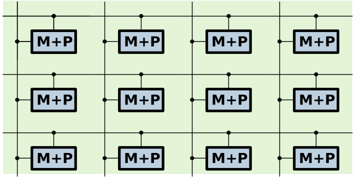
Typical Architectures for HW ANN



Exploit weight sparsity to optimize memory usage and weight placement



Use low precision/high efficiency computation along with on-chip memory storage of the weights



Integrate computation inside the memory itself, directly where the data is stored

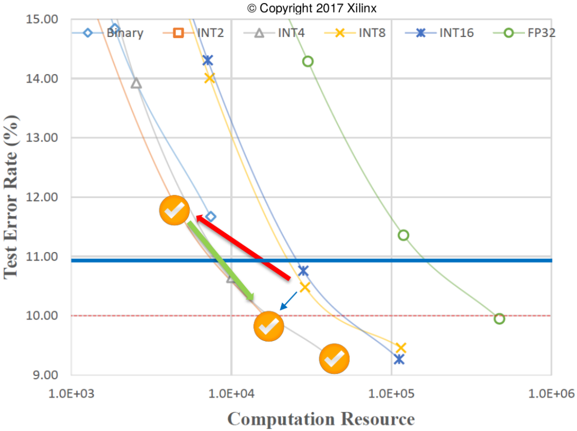
Quantization: we need it anyway!

Quantization levels and accuracy...

Just reducing precision, reduce hardware cost & increases error

Recuperate accuracy by retraining & increasing network size

1b, 2b and 4b provide pareto optimal solutions ✓



Kees Vissers, *"A Framework for Reduced Precision Neural Networks on FPGAs"*, MPSOC, 2017

Exploit Sparsity and Quantization

Custom hardware for sparse matrix-vector multiplication

Deep Compression Technique

Reduces storage requirements

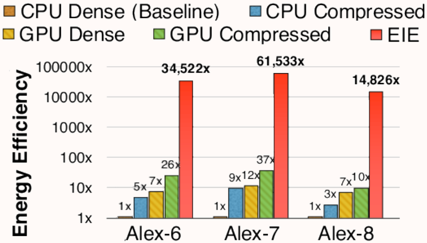
- Dedicated sparse matrix/vector representation
  ⇒ Eliminates redundant connections
- Quantizes weights

Quantization of AlexNet weights

- 256 shared weights (Conv layers) ⇒ 4 bits
- 35x of reduction (240MB ⇒ 6.9MB)

Weights stored into on-chip SRAM

⇒ 5 pJ/access (vs. 640 pJ/access off-chip DRAM)



Power efficiency

600 mW for Alexnet Fully-Connected layers

S. Han et al., *"EIE: Efficient inference engine on compressed deep neural network"*, ISCA, 2016

High weights quantization with floating-point activations

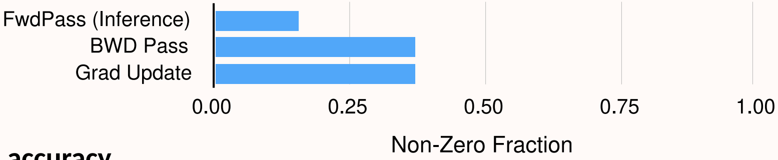
Acceleration using Low-Precision (ternary) weights

Only balanced ternary weight **are used** $\{-1, 0, +1\}$

- Floating point accumulations are kept
- Multipliers are not needed

Most of the FP operations operate on zero values

- Zero-skipping



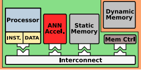
Demonstrated highest accuracy

$\Rightarrow$ 93% on the ImageNet object classification challenge

$\Rightarrow$ Divide by 3 the number of FP operations

G. Venkatesh et al., "*Accelerating Deep Convolutional Networks Using Low-Precision and Sparsity*", ICASSP, 2017

Exploit full-quantization



YodaNN: VLSI Implementation of binary-weights CNN Accelerator

Based on BinaryConnect approach

- Binary weights $\in \{-1, +1\}$
- 2's complement and multiplexers instead of multipliers
- Still full fledge adders: 12-bit activations

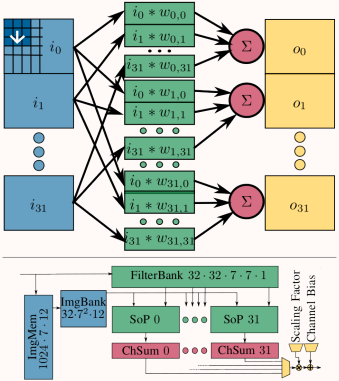
Large on-chip weights storage thanks to their size

- Latch-based standard cell memory

Flexible accelerator

- 7 kernel sizes supported

$\Rightarrow$ 61.2 TOP/s/W at 0.6V



R. Andri et al., *"YodaNN: YodaNN: An architecture for ultralow power binary-weight CNN acceleration"*, IEEE TCAD, 2017

Binary-Quantized Low-Power/Low-Area Digital Computations

FINN: Framework for building FPGA* accelerators

Mapping binarized neural networks to hardware

All values $\in \{-1, +1\}$

- Binary input activation
- Binary synapse weights
- Binary output activation

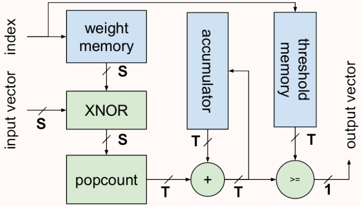
Weights kept in on-chip memory

$\Rightarrow$ Zynq-7000 FPGA technology

$\Rightarrow$ 80.1% accuracy for CIFAR-10

$\Rightarrow$ Total system power 25W

Convolution layer

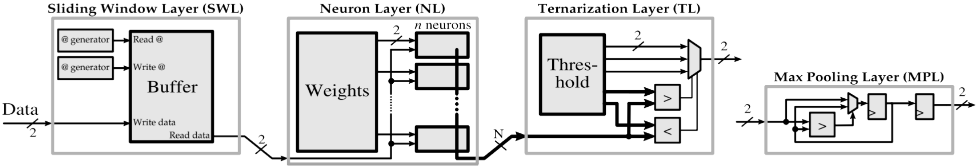


- Dot-product between input vector and row of synaptic weight matrix
- Compares result to a threshold
- Produces single-bit output

Y. Umuroglu et al., *"FINN: A Framework for Fast, Scalable Binarized Neural Network Inference"*, FPGA, 2017

[*] Field-Programmable Gate-Array: fine-grain reconfigurable hardware technology.

Ternary weights *and* ternary activations



FPGA Architecture for Ternary Neural Networks (TNN)

- Large-scale ternary CNN pipeline VGG-like (NN-64 or NN-128)
- Neuron layer (NL) $\rightarrow$ memory (ternary weights) + neurons
- Ternarization layer (TL) $\rightarrow$ ternary activations $\in \{-1, 0, +1\}$

$\Rightarrow$ Error rate 13.29% for CIFAR-10 (vs. 19.9% in FINN)

$\Rightarrow$ Virtex-7 FPGA technology (VC709, Laaaaaaarge FPGA)

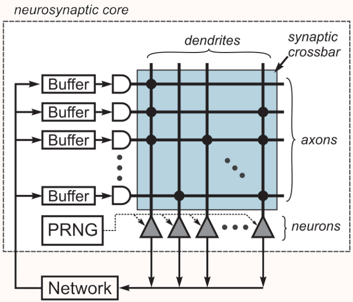
$\Rightarrow$ 1.62 TOP/s/W (vs 0.69 TOP/s/W in FINN)

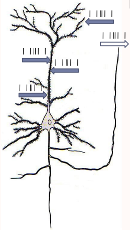
$\Rightarrow$ Throughput $>$ 60k fps

A. Prost-Boucle et al., *"High-Efficiency Convolutional Ternary Neural Networks with Custom Adder Trees and Weight Compression"*, ACM TRETS, 2018

Computations using spikes, not bits

TrueNorth: Integrated Chip for Spiking Neural Networks





Romain Brette, *Computing with spikes*

- Neurons communicate sending spikes
- Data encoded according to frequency, time, and spatial distribution of spikes

$\Rightarrow$ 4096 neuromorphic cores

$\Rightarrow$ 1 million of digital neurons

$\Rightarrow$ 256 millions of synapses

$\Rightarrow$ 46 GSOP/s/W at 65 mW

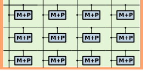
Asynchronous logic implementation

Non-Von Neumann architecture

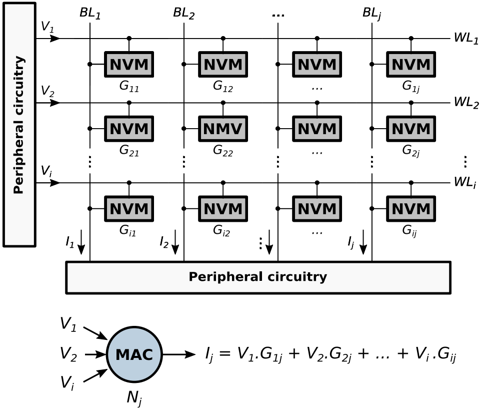
- Neuromorphic core = 256 neurons (PE) + 64k synapses (memory)
- Memory and computation physically close to each other
- Reduction of power consumption

F. Akopyan et al., "*Truenorth: Design and Tool Flow of a 65mW 1 million Neuron Programmable Neurosynaptic Chip*", TCAD, 2015

Emerging Processing-In-Memory (PIM) Approaches



MAC operations using Non Volatile Memory (NVM)



Computation accelerated by NVM arrays

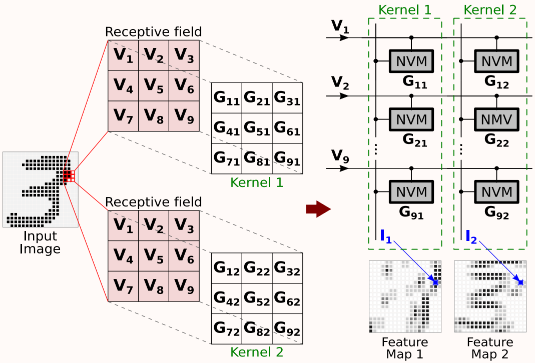
- Synaptic weights are **not** stored in external memories
  $\Rightarrow$ Zero transfers between memory and processing elements
  $\Rightarrow$ Reduction of energy consumption

Arrays of resistive RAM devices

- Resistances vary according to voltages
- No CMOS access devices but complex peripheral circuitry
- Analog, intrinsically approximate, computations

Resistive RAM Crossbar Implementations

Convolution Kernel in 12×12 Array



From receptive field to feature maps...

- Receptive field = row voltages
- Convolution kernel = column of resistive devices
- Convolution operation = column current

Interactive protocol programs kernels

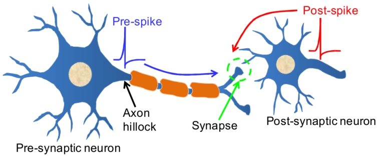
Single demonstration on digits of MNIST

L. Gao et al., *"Demonstration of Convolution Kernel Operation on Resistive Cross-Point Array"*, IEEE Electron Device Letters, 2016

P. Chi et al., *"PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation in ReRam-Based Main Memory"*, ISCA, 2016

Processing-In-Memory Approaches Using Emerging Technologies

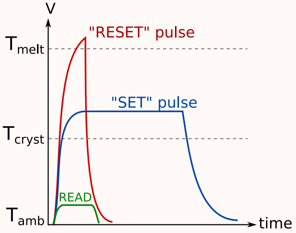
⇒ Artificial synapse using NVM

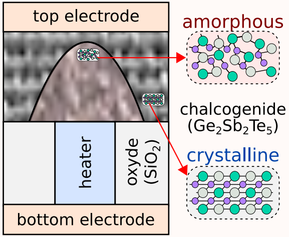


- Modeling synapses between neurons
- Input and output potentials fired between neurons (spikes)
- Synaptic connections are potentiated or depressed

Electronic Synapses Modeled by Phase Change Memory (PCM) Devices

- Programmed in different states (conductances)
- Compatible with CMOS components
- Scalable to nanometric dimensions





PCM as Synaptic Element

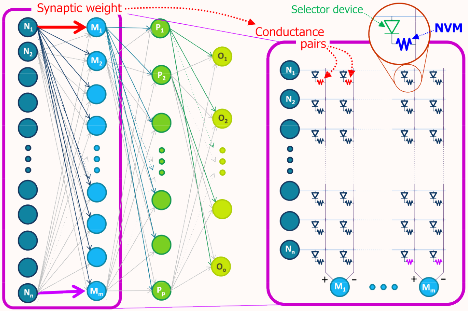
500x661 PCM Crossbar Array

Large scale implementation

- 3-layer perceptron
- 916 neurons
- 164865 synaptic connections

⇒ Accuracy: 82% (MNIST)

⇒ Low-power: at least 120x (vs. GPU)



S. Burc et al., "*Experimental Demonstration of Array-Level Learning with Phase Change Synaptic Devices*", IEDM, 2013

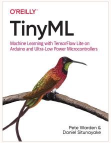
G.W. Burr et al., "*Experimental Demonstration and Tolerancing of a Large-Scale NN (165000 Synapses) using PCM as the Synaptic Weight Element*", 2015

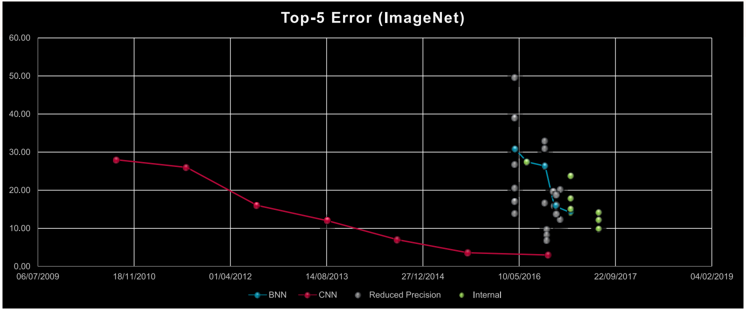
G.W. Burr et al., "*Large-Scale Neural Networks Implemented with NVM as the Synaptic Weight Element: Comparative Performance Analysis*", IEDM, 2015

Take Away

Classical digital (CMOS) architectures

- Quantization and compression is the way to go
- Few bits for weights and activations are enough in many cases
- Learning need to be changed according to bit-width
- Networks and HW architectures available today!


O'REILLY
TinyML
Machine Learning with TensorFlow Lite on Arduino and Ultra-Low Power Microcontrollers
Pete Warden & Daniel Situnayake

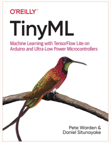


Th. Preusser, *QNNs on All Programmable Devices*, FPL 2017

Take Away

Classical digital (CMOS) architectures

- Quantization and compression is the way to go
- Few bits for weights and activations are enough in many cases
- Learning need to be changed according to bit-width
- Networks and HW architectures available today!


O'REILLY
TinyML
Machine Learning with TensorFlow Lite on Arduino and Ultra-Low Power Microcontrollers
Pete Warden &
Daniel Situnayake

Processing in-memory approaches using emerging technologies

- Few experiments of NVM crossbar array implementations at large scale
- Comparison of energy gains is difficult
- Flexibility of the NVM to match a given ANN architecture questionable
- Promising ongoing research subject