TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Institut für Computertechnik
Institute of Computer Technology

Embedded Machine Learning

Axel Jantsch

TU Wien, Vienna, Austria

September 2020

Outline

1. Motivation and Challenges
2. HW Friendly Optimizations
3. CNN Accelerator Architectures
4. Quantization

Outline

1. Motivation and Challenges
2. HW Friendly Optimizations
3. CNN Accelerator Architectures
4. Quantization

Motivation and Challenges

Why *Embedded* Machine Learning ?

Why *Embedded* Machine Learning ?

- Machine learning is a powerful method to analyze data;

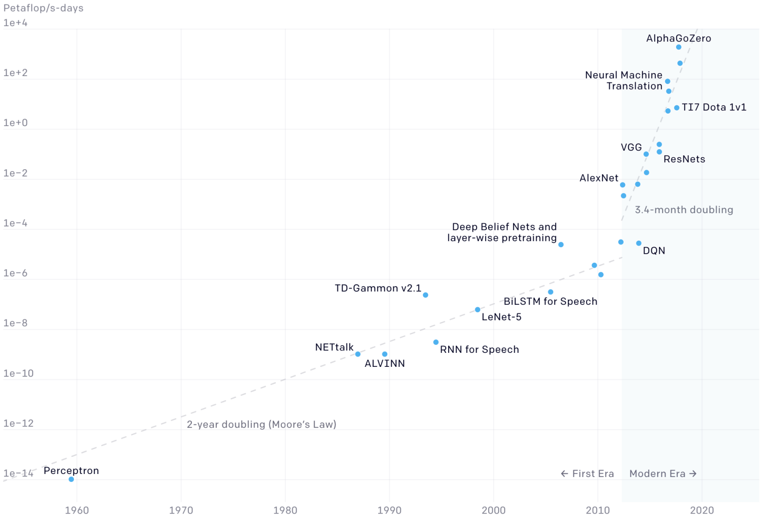
Why *Embedded* Machine Learning ?

- Machine learning is a powerful method to analyze data;
- Embedded application produce huge amounts of sensor data;

Why *Embedded* Machine Learning ?

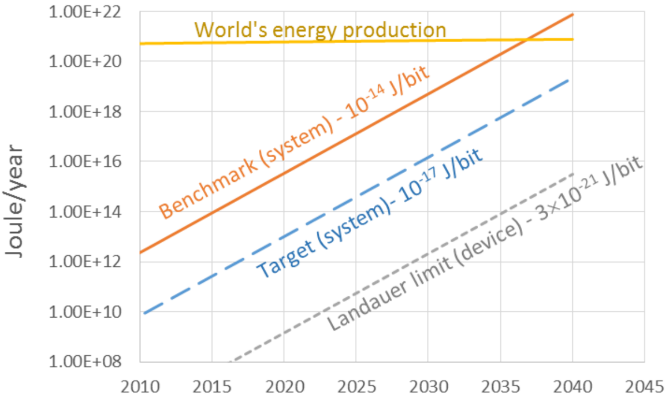
- Machine learning is a powerful method to analyze data;
- Embedded application produce huge amounts of sensor data;
- The data can or should not always be moved to central servers;

Compute Usage Trend



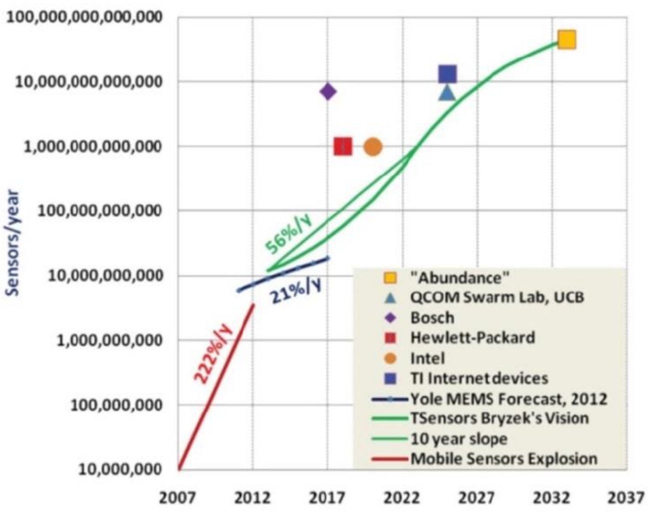
From: `https://openai.com/blog/ai-and-compute/`

Total Energy Consumption



SIA - SRC. *Rebooting the IT Revolution: A Call to Action*. Tech. rep. Semiconductor Industry Association and Semiconductor Research Corporation, Sept. 2015

Deployed Sensors



SIA - SRC. *Rebooting the IT Revolution: A Call to Action*. Tech. rep. Semiconductor Industry Association and Semiconductor Research Corporation, Sept. 2015

What is Special About “Embedded”?

- Resource limitation
- Connectivity
- Security
- Privacy

What is Special About "Embedded"?

Resource limitations

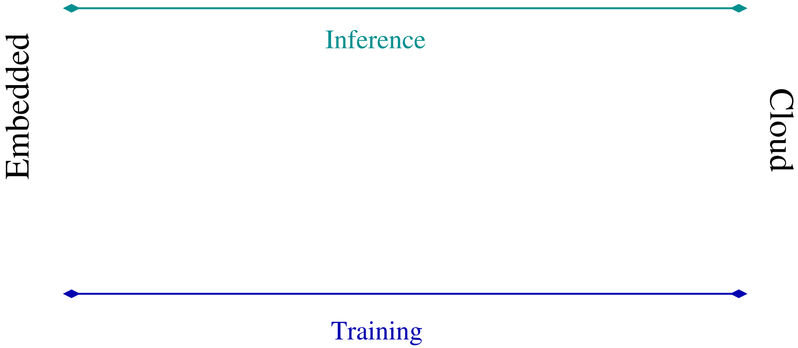
	Embedded	Server farm
Computation [flop]	$30 - 1800 \cdot 10^{12}$	$86 \cdot 10^{18}$
Memory [bit]	10^{10}	10^{15}
Power [W]	5-100	$10^3 - 10^6$
Energy [Wh]	48-1000	$200 \cdot 10^6$

Computation Embedded refers to an Nvidia Jetson Nano running 1 min and 1 hour, respectively.
Computation server refers to the computation needed for the 40 day experiment with AlphaGo Zero
Energy embedded refers to a mobile phone and to a car battery, respectively.
Energy server refers to the 40 day experiment for AlphaGo Zero.

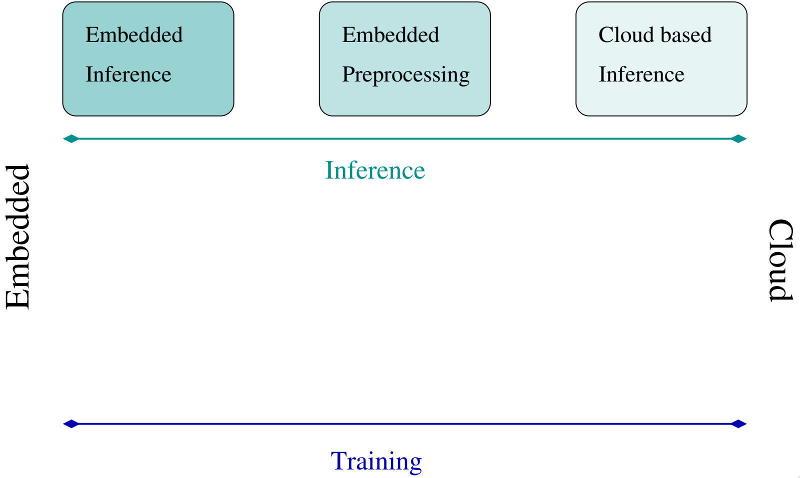
Case for Embedded ML

- Embedded inference:
  - More energy efficient
  - Bandwidth constraints
  - Latency constraints
  - Not always on-line and connected to a cloud server
  - Security
  - Privacy
- Embedded continuous learning:
  - Customization and specialization
  - Security
  - Privacy

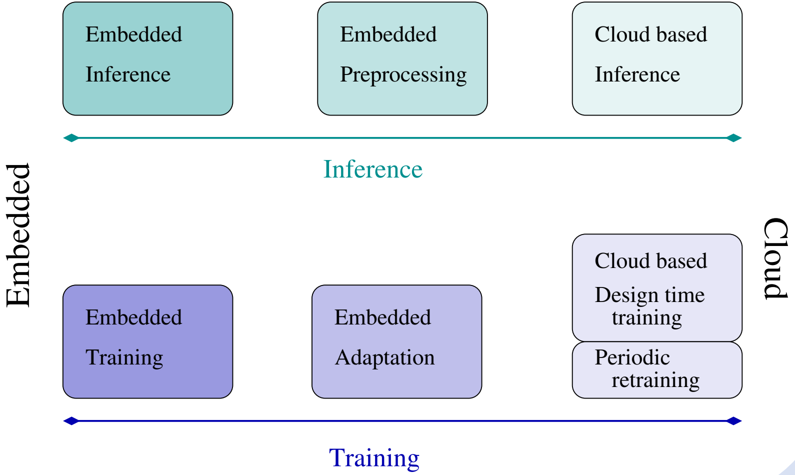
DNNs: Embedded and the Cloud



DNNs: Embedded and the Cloud

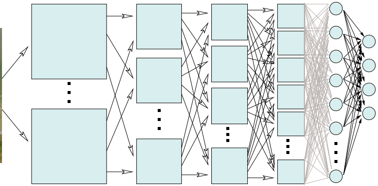


DNNs: Embedded and the Cloud



Design Space

Design Space



DNN Choices

Convolutional layers
Filter kernels
Number of filters
Pooling layers
Filter shape
Stride
Fully connected layer
Number of layers
Regularization

etc.

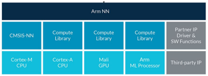
Mapping Choices

Neuron pruning
Data type selection
Approximation
Retraining
Connection pruning
Weight sparsifying
Regularization
etc.

Platform Choices

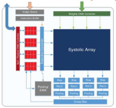
Platform Selection
Reconfiguration
Batch processing
Deep pipelining
Resource reuse
Hierarchical control
Processing unit selection
Memory allocation
Memory reuse
etc.

Intel® Vision Products with Intel® Arria® 10 FPGA



ARM NN

Xilinx DNN Processor (xDNN)



Nvidia Turing

Quiz Time

Quiz Time

1. In which year will computing consume all world-wide produced energy, if

Quiz Time

1. In which year will computing consume all world-wide produced energy, if
    a. current trends continue without major innovation in technology;
       Choices: 2027, 2032, 2037, 2042, 2047

Quiz Time

1. In which year will computing consume all world-wide produced energy, if

   a. current trends continue without major innovation in technology;
   Choices: 2027, 2032, 2037, 2042, 2047

   b. current trends continue with aggressive and major innovations in technology?

Quiz Time

1. In which year will computing consume all world-wide produced energy, if

   a. current trends continue without major innovation in technology;
      Choices: 2027, 2032, 2037, 2042, 2047

   b. current trends continue with aggressive and major innovations in technology?

Join at
slido.com
#32126

Outline

HW Friendly Optimizations

Optimization Categories

1. Minimize number of operations to be performed;

Optimization Categories

1. Minimize number of operations to be performed;
2. Simplify each operation;

Optimization Categories

1. Minimize number of operations to be performed;
2. Simplify each operation;
3. Execute the operations as efficient as possible.

HW Friendly Optimizations

- Loop reordering, unrolling, pipelining
- Tiling
- Batching
- Binarized CNNs

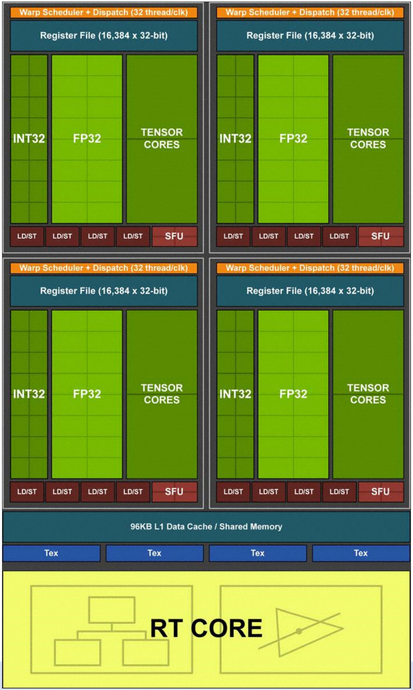
Loop Optimizations

Convolution layer algorithm:

```
for (to=0; t<M; to++) {                      // output feature map
  for (ti=0; t<N; ti++) {                    // input feature map
    for (row=0; row<R; row++) {              // row
      for (col=0; col<C; col++) {            // column
        for (i=0; i<K; i++) {                // filter
          for (j=0; j<K; j++) {
            Ofmap[to][row][col]
              += W[to][ti][i][j]
                * Ifmap[ti][S*row+i][S*col+j];
}}}}}}
```

M ... number of output feature maps
N ... number of input feature maps
R ... number of rows
C ... number of columns
K ... filter kernel size
S ... stride
W ... weight matrix

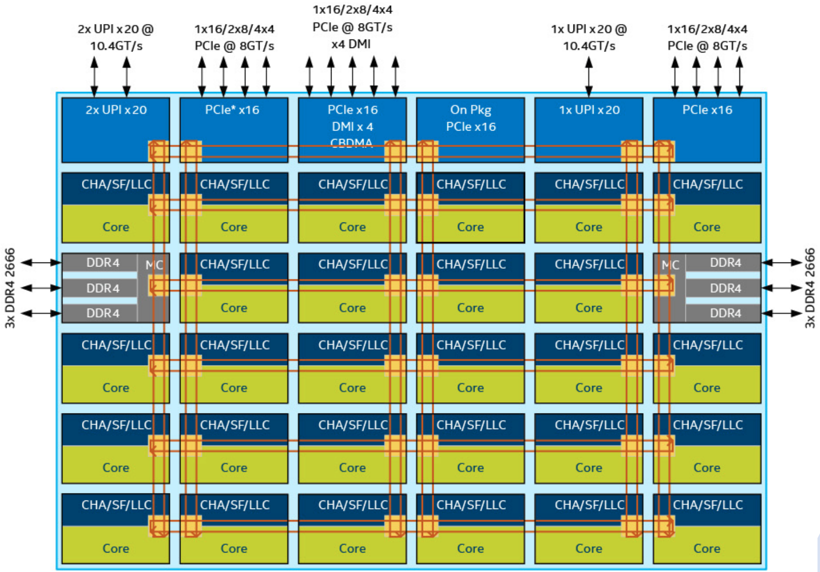

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
RT CORE
Memory Controller
L2 Cache
High-Speed Hub
NVLink – Two x8 Links



Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
LD/ST
LD/ST
LD/ST
SFU
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
LD/ST
LD/ST
LD/ST
SFU
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
LD/ST
LD/ST
LD/ST
SFU
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
LD/ST
LD/ST
LD/ST
SFU
96KB L1 Data Cache / Shared Memory
Tex
Tex
Tex
Tex
RT CORE

Nvidia Turing TU102

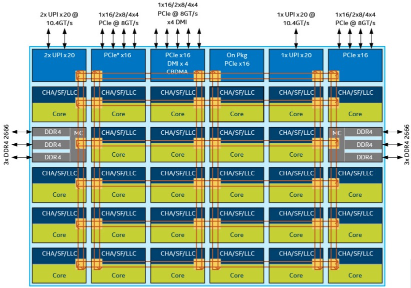
Process (nm)	12
Transistors (billion)	18.6
Die size (mm^2)	754
Streaming Multiprocessors (SM)	72
CUDA Cores	4608 (64/SM)
Tensor Cores	576 (8/SM)
RT Cores	72
Clock (MHz)	$\leq$ 1500
CUDA TFlops (FP32)	13.8
L1 Cache (MB)	6.912
L2 Cache (MB)	6
Bus width	384
Power (W)	200-250
Bandwidth (GB/s)	672


2x UPI x20 @ 10.4GT/s
1x16/2x8/4x4 PCIe @ 8GT/s
1x16/2x8/4x4 PCIe @ 8GT/s x4 DMI
1x UPI x20 @ 10.4GT/s
1x16/2x8/4x4 PCIe @ 8GT/s
2x UPI x20
PCIe* x16
PCIe x16 DMI x 4 CBDMA
On Pkg PCIe x16
1x UPI x20
PCIe x16
CHA/SF/LLC
Core
DDR4
MC
3x DDR4 2666

Intel Skylake Server Architecture

28 cores; 30 tiles, 14nm, 694 mm2

- L0 μOP cache:
  - 1536 μOPs/core,
  - 8-way set associative
  - 32 sets, 6-μOP line size
- L1 I-Cache:
  - 32 KiB/core,
  - 8-way set associative
  - 64 sets, 64 B line size
- L1 D-Cache:
  - 32 KiB/core,
  - 8-way set associative
  - 64 sets, 64 B line size
  - 4 - 5 cycles latency
  - Write-back policy
- L2 Cache:
  - 1 MiB/core,
  - 16-way set associative
  - 64 B line size
  - Write-back policy
  - 14 cycles latency
- L3 Cache:
  - 1.375 MiB/core,
  - 11-way set associative,
  - shared across all cores
  - 2,048 sets, 64 B line size
  - Write-back policy
  - 50-70 cycles latency



Loop Optimizations

Loop reordering to improve cache efficiency;

Loop unrolling to improve parallelism;

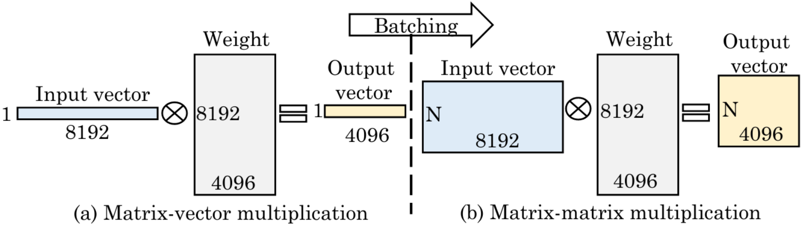
Loop pipelining to improve parallelism.

Loop Tiling

```
for (to=0; t<M; to++)   {         // output feature map
  for (ti=0; t<N; ti++) {         // input feature map
    for (row=0; row<R; row+=Tr) { // tiled row loop
      for (col=0; col<C; col++) { // column
        for (trr=row; trr<min(R,row+Tr); trr++) {
          for (i=0; i<K; i++) {   // filter
            for (i=0; i<K; i++) {
              Ofmap[to][trr][col]
                += W[to][ti][i][j]
                   * Ifmap[ti][S*trr+i][S*col+j];
            }}}}}}}
```

For efficient use of caches.

Batching



- Reuse of weights;
- Improves throughput;
- Increases latency.

Binarized CNNs (BNN)

- Weights and internal computation are represented as 1b binary numbers;
- Instead of MAC operations, BNNs use XOR and bit-count;
- Attractive for HW and FPGAs

$$\begin{bmatrix} -1 & +1 & +1 \end{bmatrix} \otimes \begin{bmatrix} -1 & -1 & +1 \\ +1 & -1 & +1 \\ +1 & -1 & +1 \end{bmatrix} \equiv \begin{matrix} (-1.-1) & (-1.-1) & (-1.+1) \\ +(+1.+1) & +(+1.-1) & +(+1.+1) \\ +(+1.+1) & +(+1.-1) & +(+1.+1) \end{matrix} \equiv \begin{bmatrix} 3 & -1 & +1 \end{bmatrix}$$

(a) An example of binarized MM

$$\begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \equiv \begin{pmatrix} \mathrm{BCNT(XNOR(011,011))} \\ \mathrm{BCNT(XNOR(011,000))} \\ \mathrm{BCNT(XNOR(011,111))} \end{pmatrix}^T \equiv \begin{bmatrix} 3 & -1 & +1 \end{bmatrix}$$

(b) Binarized MM using XNOR and BCNT. -1 is represented using 0.

BCNT= OneCount-ZeroCount

IN	Computation	OUT
000	-1-1-1= -3	101
001	-1-1+1 = -1	111
010	-1+1-1= -1	111
011	-1+1+1= +1	001
100	+1-1-1= -1	111
101	+1-1+1= +1	001
110	+1+1-1= +1	001
111	+1+1+1= +3	011

(c) BCNT using a lookup table (OUT is in 2's complement form)

Quiz Time

Join at
slido.com
#32126

Quiz Time

2. Which optimization method is most promising in improving ML compute efficiency:

Join at
slido.com
#32126

Quiz Time

2. Which optimization method is most promising in improving ML compute efficiency:

   a. Optimize network to minimize number of computations,

Join at
slido.com
#32126

Quiz Time

2. Which optimization method is most promising in improving ML compute efficiency:

   a. Optimize network to minimize number of computations,

   b. Improve processing element and processing datapath architecture,

Join at
slido.com
#32126

Quiz Time

2. Which optimization method is most promising in improving ML compute efficiency:

   a. Optimize network to minimize number of computations,

   b. Improve processing element and processing datapath architecture,

   c. Improve memory architecture,

Join at
slido.com
#32126

Quiz Time

2 Which optimization method is most promising in improving ML compute efficiency:

a Optimize network to minimize number of computations,

b Improve processing element and processing datapath architecture,

c Improve memory architecture,

d Optimize mapping of network onto target architecture ?

Join at
slido.com
#32126

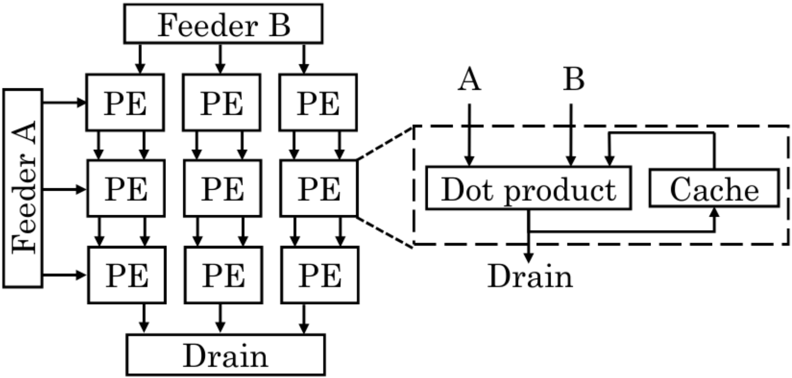
Outline

CNN Accelerator Architectures

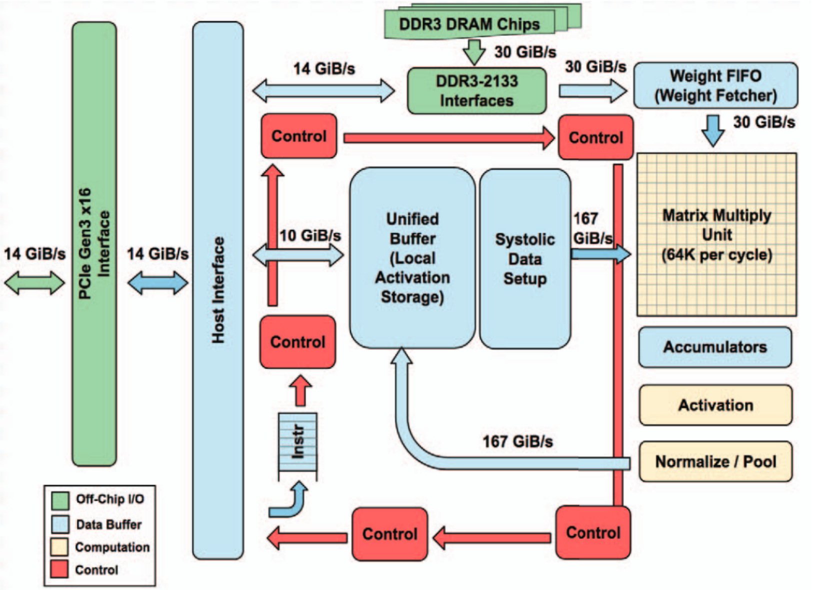
CNN Accelerator Architectures

- Systolic array architecture
- In-memory computing

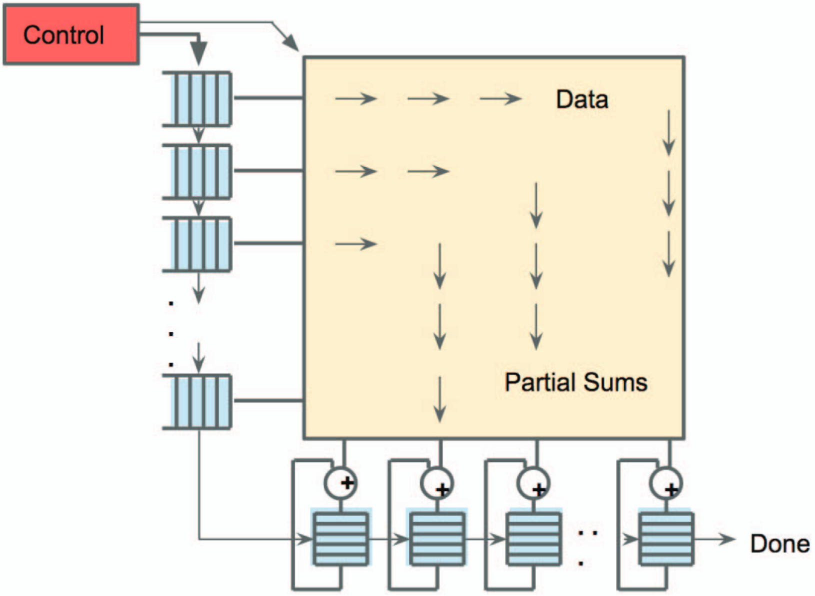
Systolic Arrays



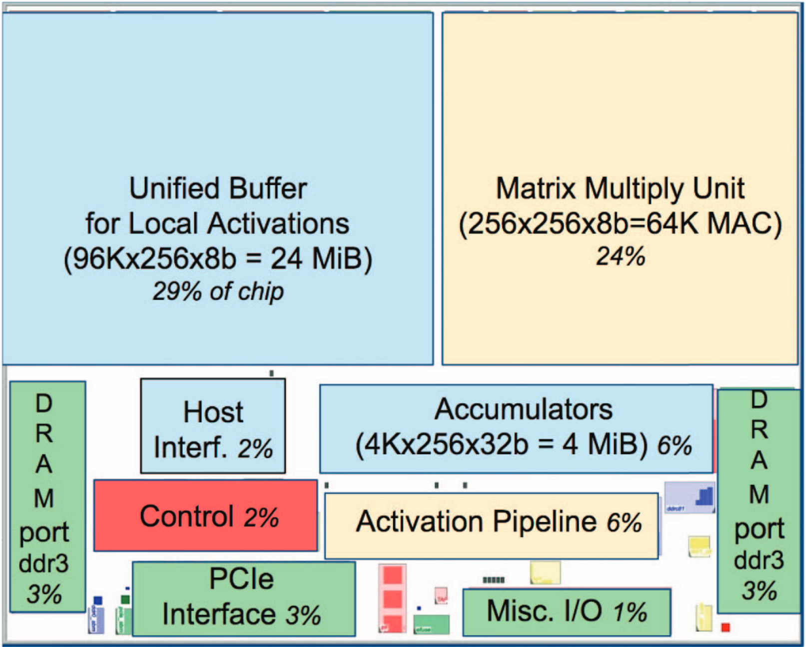
Tensor Flow Unit (TPU)



Tensor Flow Unit (TPU)



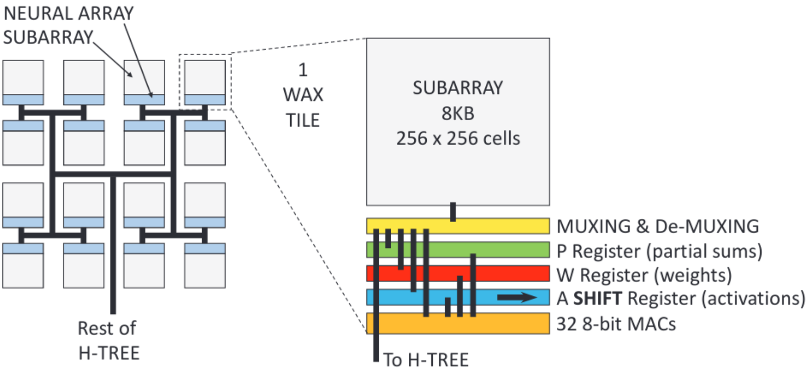
Tensor Flow Unit (TPU)



In-memory Computing

- Storage capacity and memory access bandwidth and latency dominate DNNs.
- Avoid moving data.
- Distribute the MAC units in the memory architecture.

Wire Aware Accelerator (WAX)



Sumanth Gudaparthi et al. "Wire-Aware Architecture and Dataflow for CNN Accelerators". In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO '52. Columbus, OH, USA: Association for Computing Machinery, 2019

Quiz Time

Join at
slido.com
#32126

Quiz Time

3. For optimizing CNN execution, is it more effective

Join at
slido.com
#32126

Quiz Time

❸ For optimizing CNN execution, is it more effective

a to re-use (and keep on-chip) input data,

Join at
slido.com
#32126

Quiz Time

3. For optimizing CNN execution, is it more effective
   - a to re-use (and keep on-chip) input data,
   - b to re-use weights,

Quiz Time

❸ For optimizing CNN execution, is it more effective

- a to re-use (and keep on-chip) input data,
- b to re-use weights,
- c to re-use intermediate data ?

Join at
slido.com
#32126

Outline

Quantization

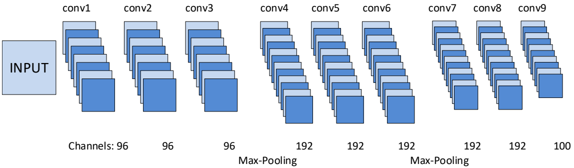
Quantization - Regularization

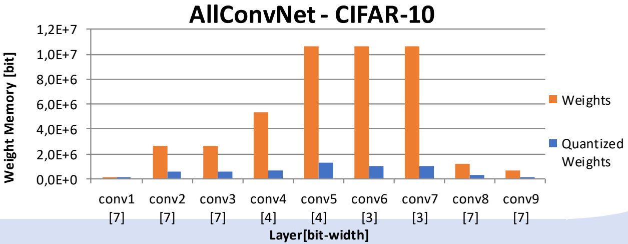
- Using small bit width for weights saves memory, bandwidth and computation;
- Bit width can be different for different layers of the DNN;
- Quantization scheme: Dynamic fixed point, power of 2;
- Retraining after quantization recovers accuracy losses: Regularization;
- Not all weights are equal: Weighted regularization.

Matthias Wess, Sai Manoj Pudukotai Dinakarrao, and Axel Jantsch. "Weighted Quantization-Regularization in DNNs for Weight Memory Minimization towards HW Implementation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.10 (Oct. 2018)

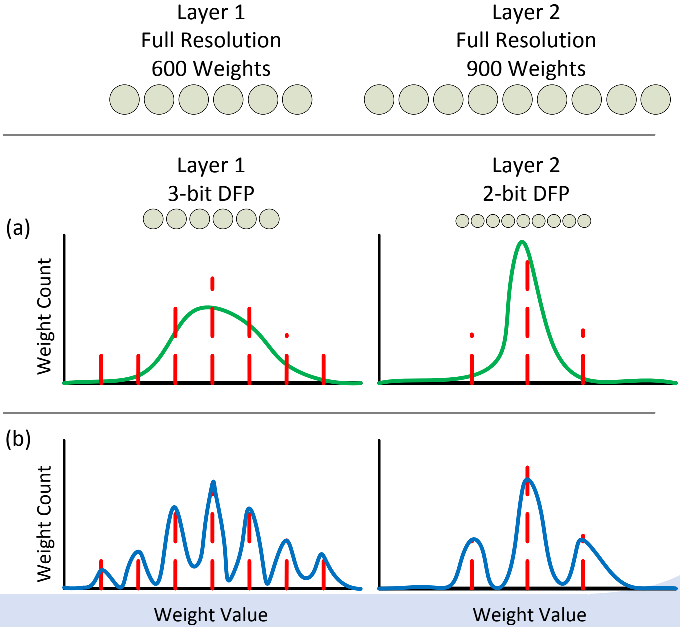
Quantization - Motivation

- DNN quantization
  - Reduces data movement
  - Reduces logic energy
- Layerwise bit-width optimization

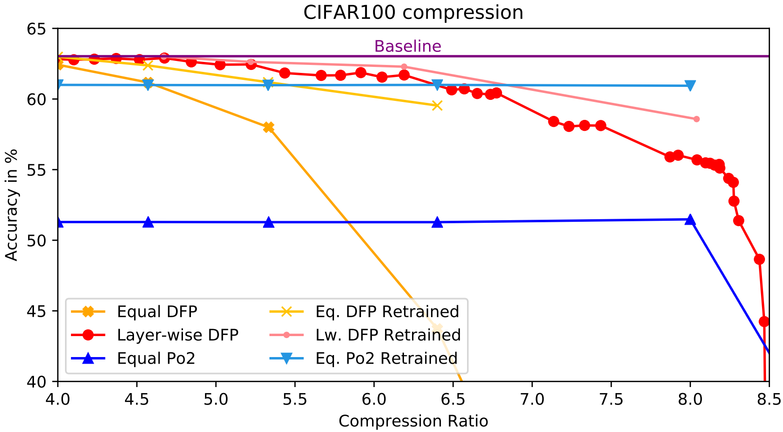




Quantization - Motivation



CIFAR-100


CIFAR100 compression
Baseline
Accuracy in %
Compression Ratio
65
60
55
50
45
40
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
Equal DFP
Layer-wise DFP
Equal Po2
Eq. DFP Retrained
Lw. DFP Retrained
Eq. Po2 Retrained

Quantization - Conclusion

- Dynamic Fixed Point is an effective alternative to integer or floating point representation;
- Different layers require different precision;
- Retraining adjusts the network to the available weight values;
- Weight memory reduction of 4-8x is common;
- Reduced weight precision reduces weight memory and cost of operation.

Summary

Summary

- Embedded Machine Learning has many applications;
  - Bandwidth limitations;
  - Delay constraints;
  - Privacy;
  - Security;

Summary

- Embedded Machine Learning has many applications;
  - Bandwidth limitations;
  - Delay constraints;
  - Privacy;
  - Security;
- There are distinct challenges:
  - Limited resources;
  - Specialized HW platforms;
  - Huge design space for optimization and mapping.

References I

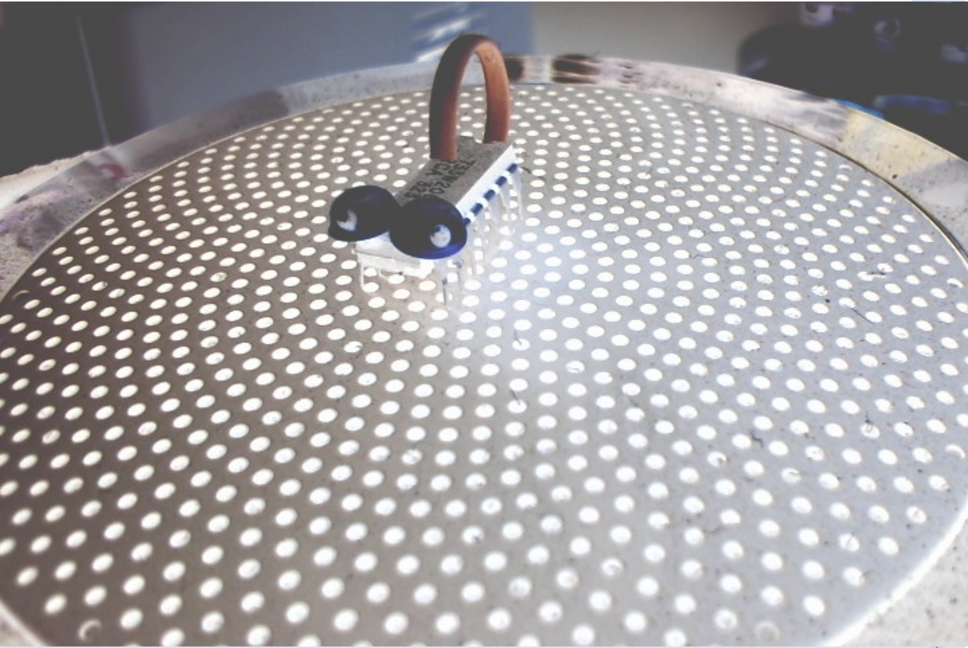
Sumanth Gudaparthi et al. "Wire-Aware Architecture and Dataflow for CNN Accelerators". In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO '52. Columbus, OH, USA: Association for Computing Machinery, 2019.

N. P. Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. 2017, pp. 1–12.

SIA - SRC. *Rebooting the IT Revolution: A Call to Action*. Tech. rep. Semiconductor Industry Association and Semiconductor Research Corporation, Sept. 2015.

Matthias Wess, Sai Manoj Pudukotai Dinakarrao, and Axel Jantsch. "Weighted Quantization-Regularization in DNNs for Weight Memory Minimization towards HW Implementation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.10 (Oct. 2018).

Yu Wang, Gu-Yeon Wei, and David Brooks. "Benchmarking TPU, GPU, and CPU Platforms for Deep Learning". In: *CoRR* abs/1907.10701 (2019). arXiv: `1907.10701`.




TECHNISCHE HOCHSCHULE