Design objectives for deep NN (DNN) hardware (HW)

Intelligent Architectures: 5LIL0

March 21, 2025

Dr. Federico Corradi

Department of Electrical Engineering, Electronic Systems Group

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Outline

Summary: Combinational and Sequential Logic in SystemVerilog

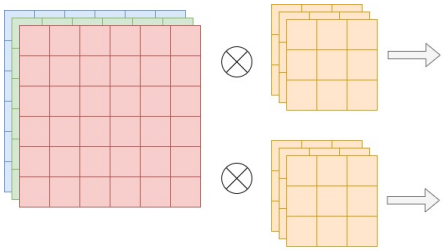
Combinational Logic (`always_comb`, `assign`)

- **Continuous Assignment**: Use `assign` for direct combinational logic.
- `always_comb`: infers the sensitivity list and is used for describing combinational logic procedurally.
- **Blocking Assignment (=)**: Used inside `always_comb` for immediate evaluation.
- Example:
  - `assign y = a & b;` (Continuous)

Sequential Logic (`always_ff`, <=)

- Use `always_ff` to model clocked (**sequential**) logic like flip-flops and registers.
- **Non-blocking Assignment (<=)** is used to model parallel updates of state.
- Ensures clarity and avoids issues like race conditions in sequential logic.
- Example:
  - `always_ff @(posedge clk) q <= d;` (Flip-flop behavior)

Executing CNN: example of a CNN layer

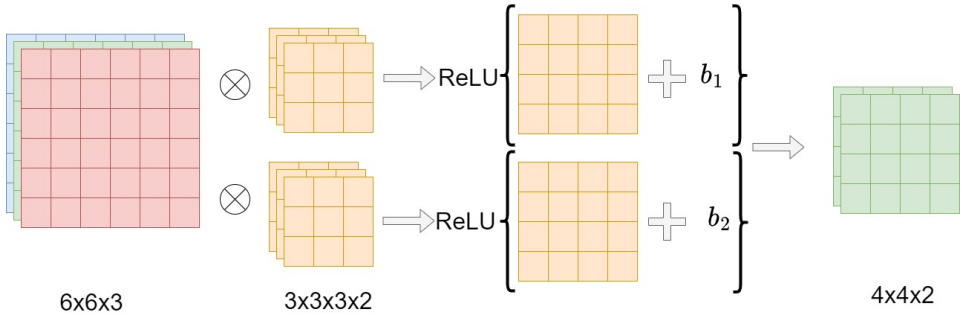


6x6x3

3x3x3x2

?x?x?

Executing CNN: example of a CNN layer



$$O[z][u][x][y] = G\left(B[u] + \sum_{k=0}^{C-1}\sum_{i=0}^{S-1}\sum_{j=0}^{R-1} I[z][k][U_x+1][U_y+j] \times W[u][k][i][j]\right) \quad (1)$$

U→stride, B→bias, C→i fmpas,M→o fmaps, N→batch, RxS→ filter

Outline

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.
- Today, we will discuss:

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.
- Today, we will discuss:
  - What are the key metrics that should be measured and compared?

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.
- Today, we will discuss:
  - What are the key metrics that should be measured and compared?
  - What are the challenges towards achieving these metrics?

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.
- Today, we will discuss:
  - What are the key metrics that should be measured and compared?
  - What are the challenges towards achieving these metrics?
  - What are the design considerations and tradeoffs?

The goal of this lesson

- In this part of the course, we will focus on approaches for the design of efficient DNN hardware processors beyond CPUs and GPUs.
- Today, we will discuss:
  - What are the key metrics that should be measured and compared?
  - What are the challenges towards achieving these metrics?
  - What are the design considerations and tradeoffs?
- We will focus on inference, but many concepts also apply to training.

Outline

Training vs Inference

- **Training**: Tuning parameters using training data
  1. Backpropagation using stochastic gradient descent is the most popular algorithm
  2. Training in data centers and distributing trained models is a common workflow
  3. Because training algorithm changes rapidly, GPU cluster is the most popular hardware (Low demand for application-specific accelerators)

Training vs Inference

- **Training**: Tuning parameters using training data
  1. Backpropagation using stochastic gradient descent is the most popular algorithm
  2. Training in data centers and distributing trained models is a common workflow
  3. Because training algorithm changes rapidly, GPU cluster is the most popular hardware (Low demand for application-specific accelerators)

- **Inference**: Determining class of a new input data
  1. Using a trained model, determine class of a new input data
  2. Inference usually occurs in the real-world (i.e., not in datacenters)
  3. Low-latency and power-efficiency is required (High demand for application specific accelerators)

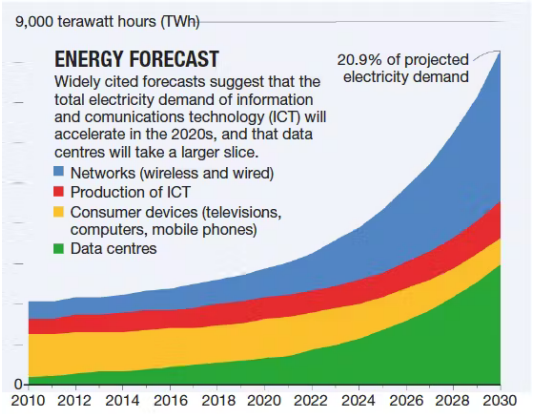
Processing at the “Edge” instead of the “Cloud”

Communication

Privacy

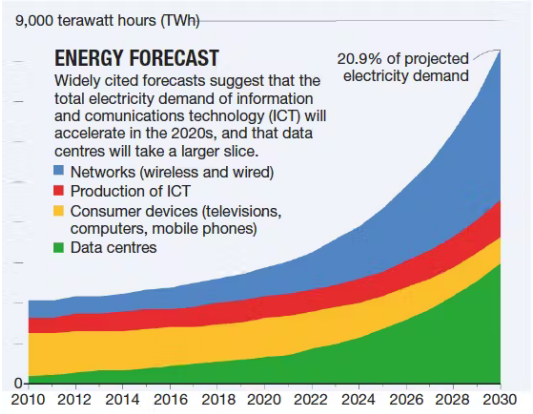
Latency

Processing at the “Edge” instead of the “Cloud”

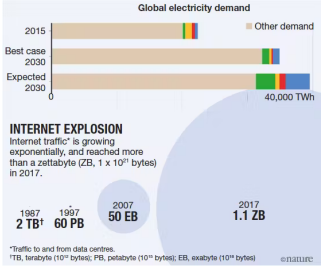


ICT specialist Andres Andrea
*[Masanet E, Science, 2020]

Processing at the “Edge” instead of the “Cloud”



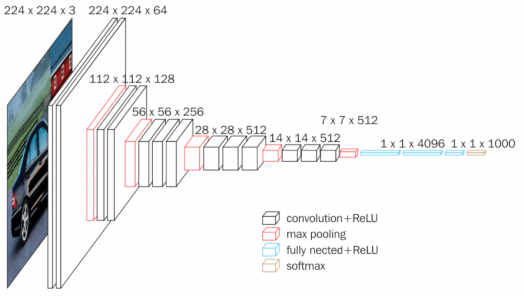
ICT specialist Andres Andrea
*[Masanet E, Science, 2020]



Efficiency is key

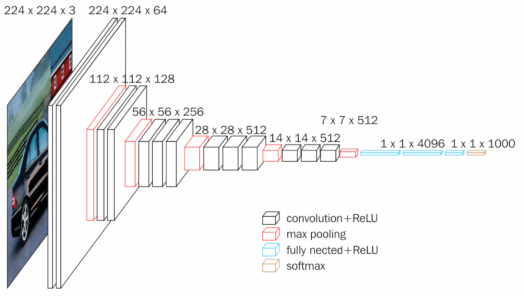
Global data center electricity use was estimated at 205 terawatt-hours (TWh) in 2018, with a 6% increase since 2010. In contrast, global data center compute instances rose by approx. 550%X during the same period, indicating significant efficiency improvements*.

The Need for Neural Network Accelerators: VGG16



Example: VGG16

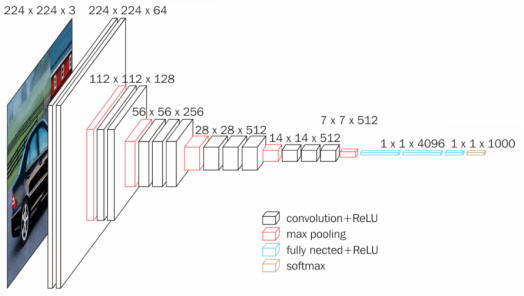
The Need for Neural Network Accelerators: VGG16



Example: VGG16

- 138 million weights, 15.5G (10^9) MACs

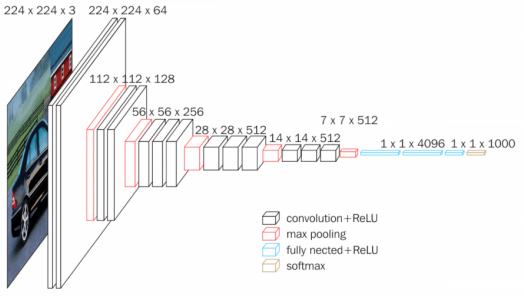
The Need for Neural Network Accelerators: VGG16



Example: VGG16

- 138 million weights, 15.5G (10^9) MACs
- CPU @ 3 GHz, 1 IPC, (3GOPS): 5+ seconds per image

The Need for Neural Network Accelerators: VGG16

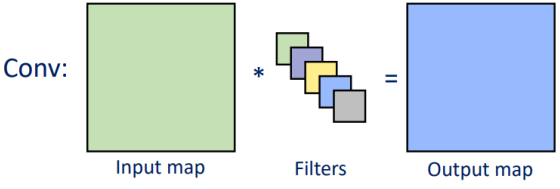


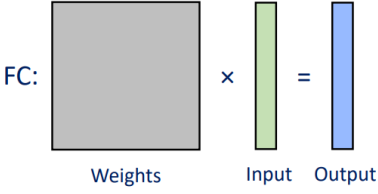
Example: VGG16

- 138 million weights, 15.5G (10^9) MACs
- CPU @ 3 GHz, 1 IPC, (3GOPS): 5+ seconds per image
- Significant power consumption! Optimistically assuming 3 GOPS/thread at 8 threads using 100 W, 0.24 GOPS/W

Two Major Layers

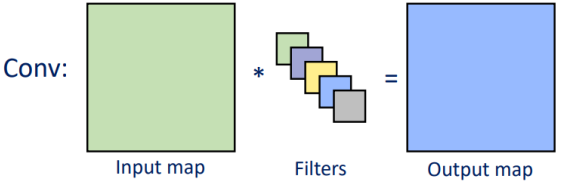
- **Convolution Layer**:

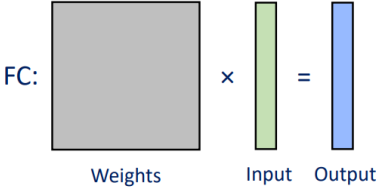




Two Major Layers

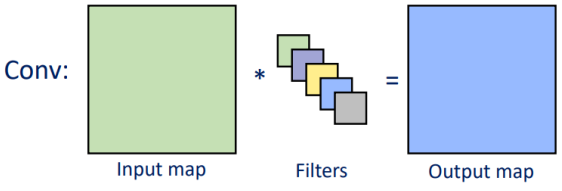
- **Convolution Layer**:
  1. Many small (1x1, 3x3, 11x11, . . .) filters

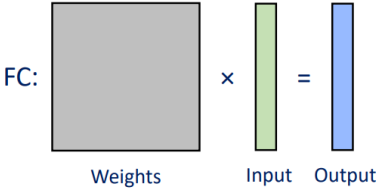




Two Major Layers

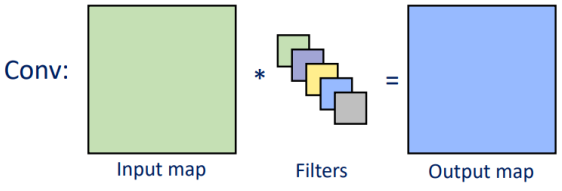
- **Convolution Layer**:
  1. Many small (1x1, 3x3, 11x11, . . .) filters
  2. Over 90% of the multiply accumulate (MAC) operations in a typical model

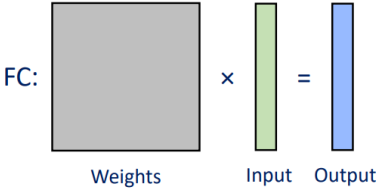




Two Major Layers

- **Convolution Layer**:
  1. Many small (1x1, 3x3, 11x11, . . .) filters
  2. Over 90% of the multiply accumulate (MAC) operations in a typical model
  3. Small number of weights per filter, relatively small number in total vs. FC

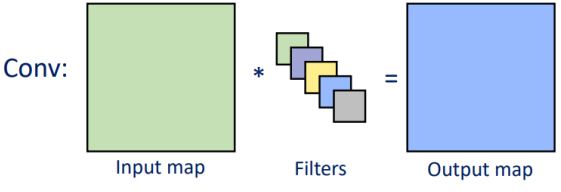


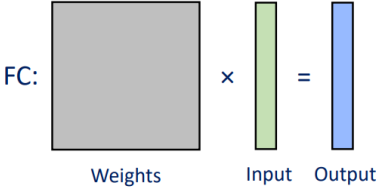


Two Major Layers

- **Convolution Layer**:
  1. Many small (1x1, 3x3, 11x11, . . .) filters
  2. Over 90% of the multiply accumulate (MAC) operations in a typical model
  3. Small number of weights per filter, relatively small number in total vs. FC

- **Fully-Connected Layer**:

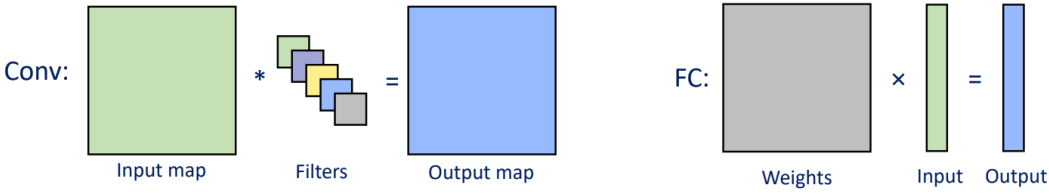




Two Major Layers

- **Convolution Layer**:
  1. Many small (1x1, 3x3, 11x11, . . .) filters
  2. Over 90% of the multiply accumulate (MAC) operations in a typical model
  3. Small number of weights per filter, relatively small number in total vs. FC

- **Fully-Connected Layer**:
  1. N-to-N connection between all neurons, large number of weights



Introduction to key HW metrics

key HW metrics

Introduction to key HW metrics

key HW metrics

- Latency (time per inference) $\rightarrow$ throughput (inferences per second)

Introduction to key HW metrics

key HW metrics

- Latency (time per inference) $\rightarrow$ throughput (inferences per second)
- Battery lifetime $\rightarrow$ energy efficiency operations per second per Watt (i.e., TOPS/W)

Introduction to key HW metrics

key HW metrics

- Latency (time per inference) $\rightarrow$ throughput (inferences per second)
- Battery lifetime $\rightarrow$ energy efficiency operations per second per Watt (i.e., TOPS/W)
- Memory usage (weight/intermediate data)

What is needed to put DNN in edge devices?

Assumptions

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 10W!

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 10W!

Conclusions

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 10W!

Conclusions

- We need 180 frames/second (6 cameras x 30 frames/second).

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 10W!

Conclusions

- We need 180 frames/second (6 cameras x 30 frames/second).
- To fit in 10W power budget, we need to divide the total computational capacity (in GOPs) by the power budget (in watts)

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

- 6 full HD cameras @ 30 fps, 1920x1080x3 (RGB)
- Resnet-50/frame (550 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 10W!

Conclusions

- We need 180 frames/second (6 cameras x 30 frames/second).
- To fit in 10W power budget, we need to divide the total computational capacity (in GOPs) by the power budget (in watts)
- (550 GOPS/frame x 180 inferences/second) / 10W = 9900 GOPS/W $\sim$ 10TOPs/W)

[Convnet burden, Albanie]

What is needed to put DNN in edge devices?

Assumptions

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)
- Resnet-50/frame (200 GOPs/frame - 32-bit batch size of 1, underestimation!)

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)
- Resnet-50/frame (200 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 100mW!

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)
- Resnet-50/frame (200 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 100mW!

Conclusions

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)
- Resnet-50/frame (200 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 100mW!

Conclusions

- We need 60 inferences/second (2 cameras x 30 frames/second).

What is needed to put DNN in edge devices?

Assumptions

- 2 stereo cameras @ 30 fps, 1280x720x3 (RGB)
- Resnet-50/frame (200 GOPs/frame - 32-bit batch size of 1, underestimation!)
- I only have 100mW!

Conclusions

- We need 60 inferences/second (2 cameras x 30 frames/second).
- (200 GOPS/frame x 60 inferences/second) / 0.1W = 120 TOPS/W

Summary: DNN Deployment in Edge Devices

Key Assumptions

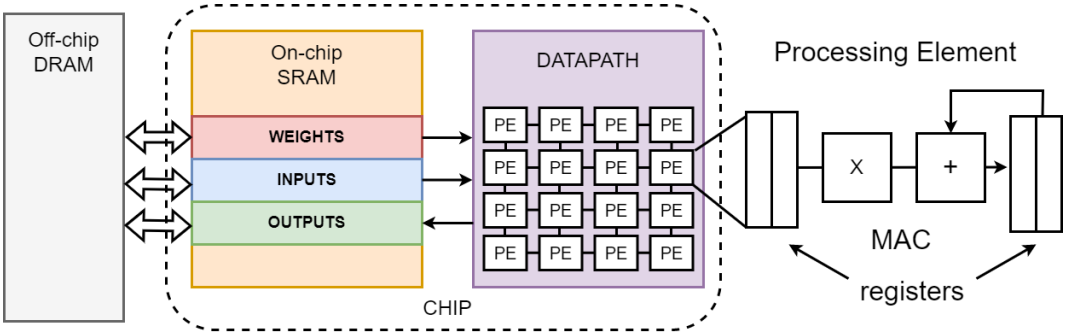
- Scenario 1: 6 full HD cameras @ 30 fps (1920x1080x3 RGB)
- Scenario 2: 2 stereo cameras @ 30 fps (1280x720x3 RGB)
- Processing: ResNet-50/frame
- Power Constraints: 10W (Scenario 1), 100mW (Scenario 2)

Key Conclusions

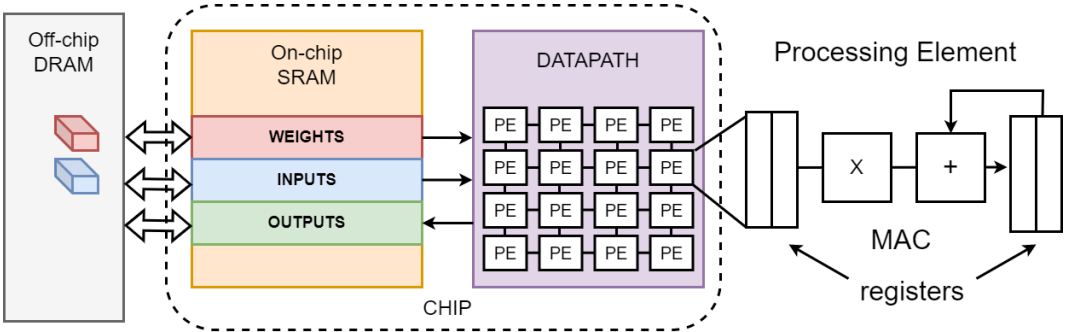
- Scenario 1: Requires 180 inferences/sec
  Computational efficiency needed: **10 TOPS/W**
- Scenario 2: Requires 60 inferences/sec
  Computational efficiency needed: **120 TOPS/W**

Outline

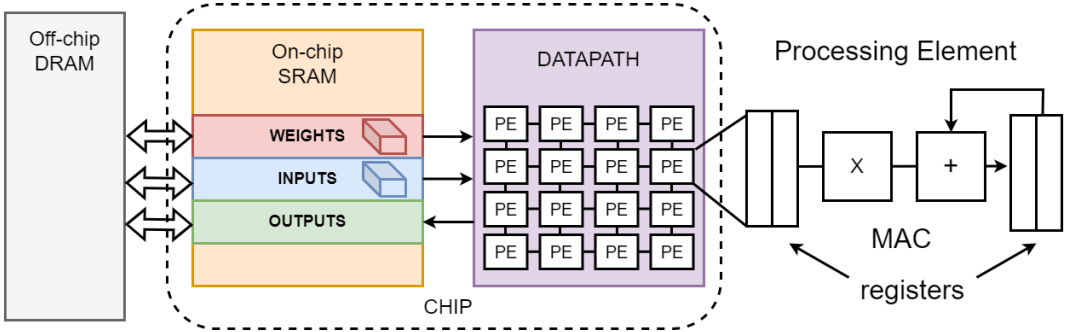
Energy Efficient DNN processors



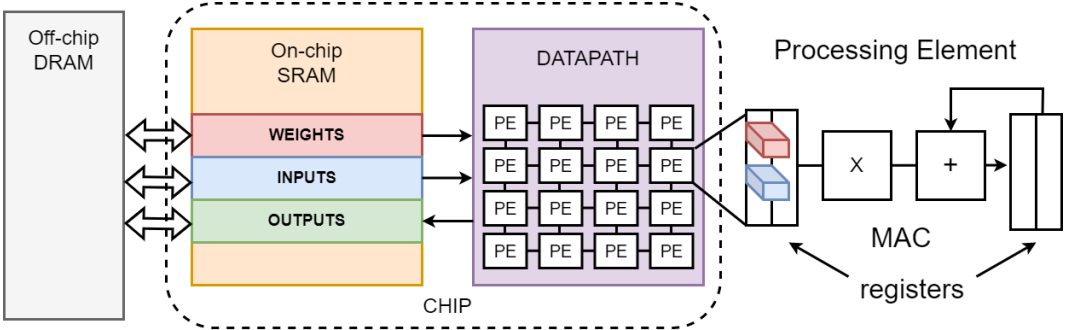
Energy Efficient DNN processors



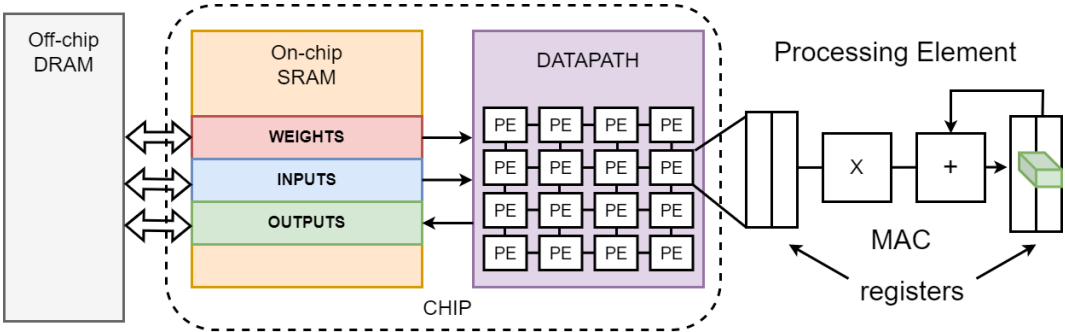
Energy Efficient DNN processors



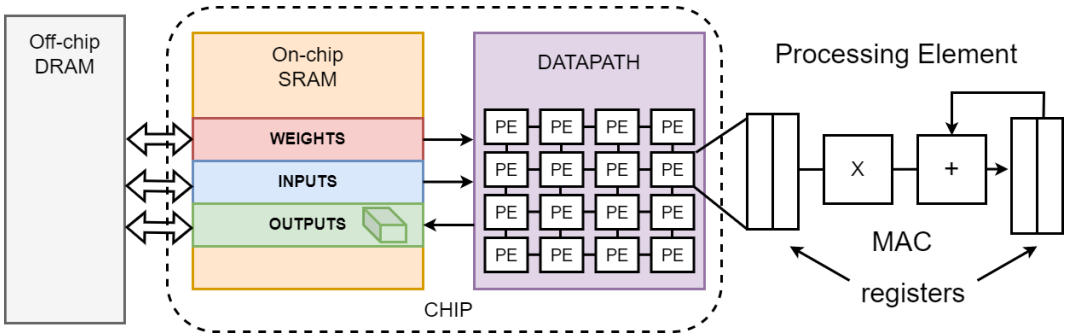
Energy Efficient DNN processors



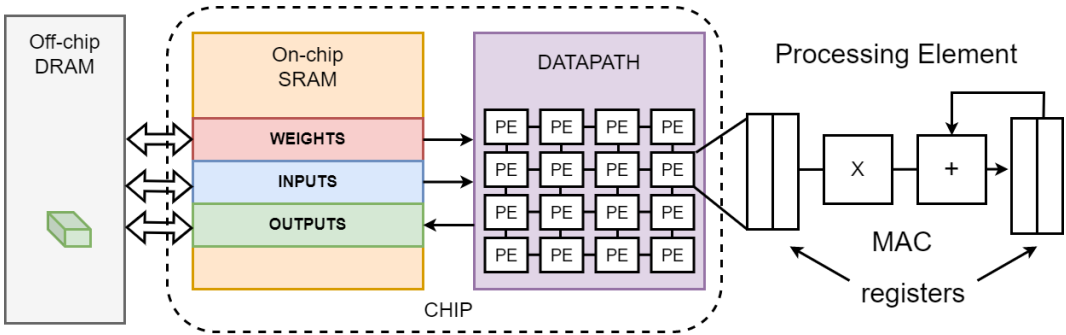
Energy Efficient DNN processors



Energy Efficient DNN processors

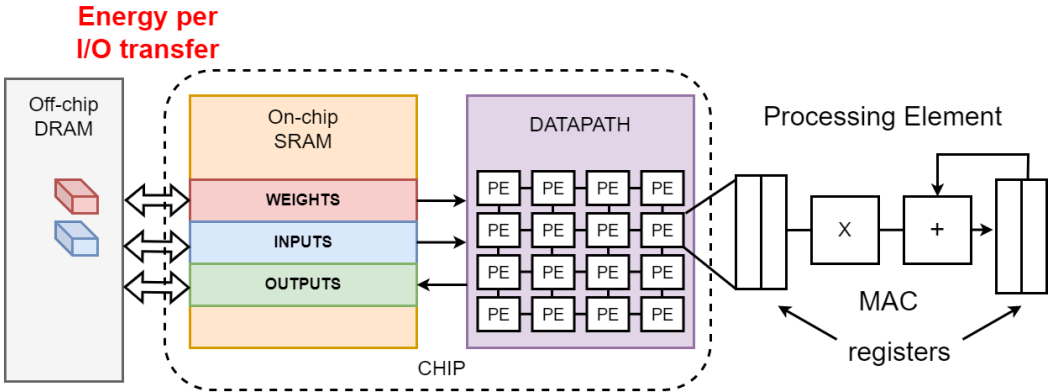


Energy Efficient DNN processors



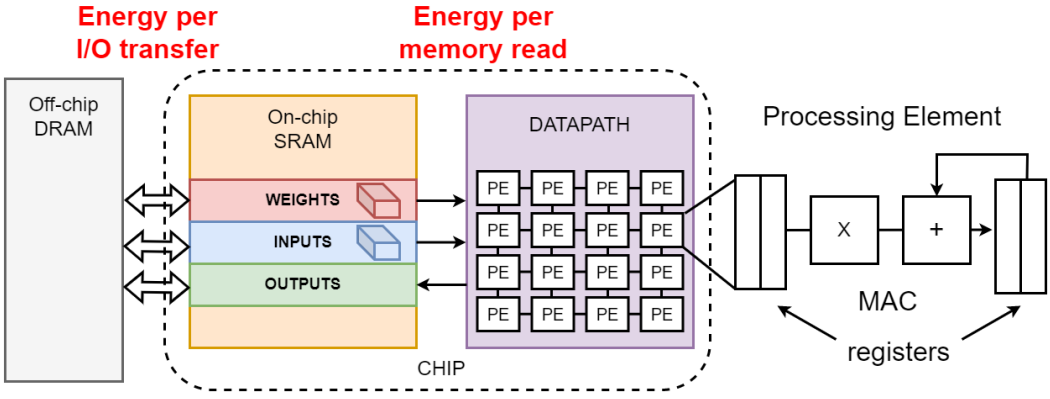
Energy Efficient DNN processors

ENERGY EFFICIENCY: Joule/op



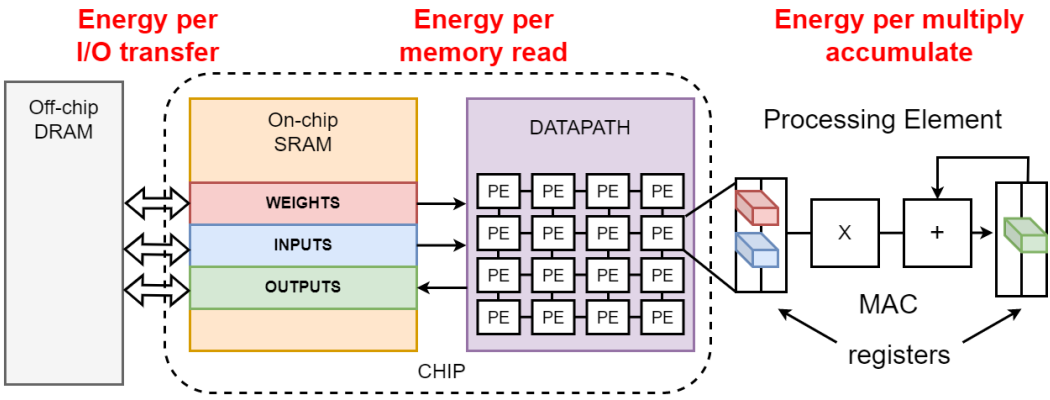
Energy Efficient DNN processors

ENERGY EFFICIENCY: Joule/op



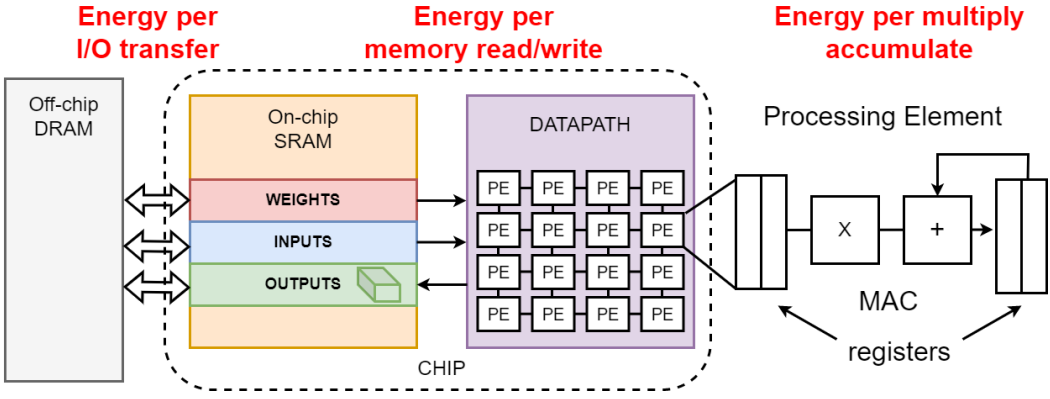
Energy Efficient DNN processors

ENERGY EFFICIENCY: Joule/op



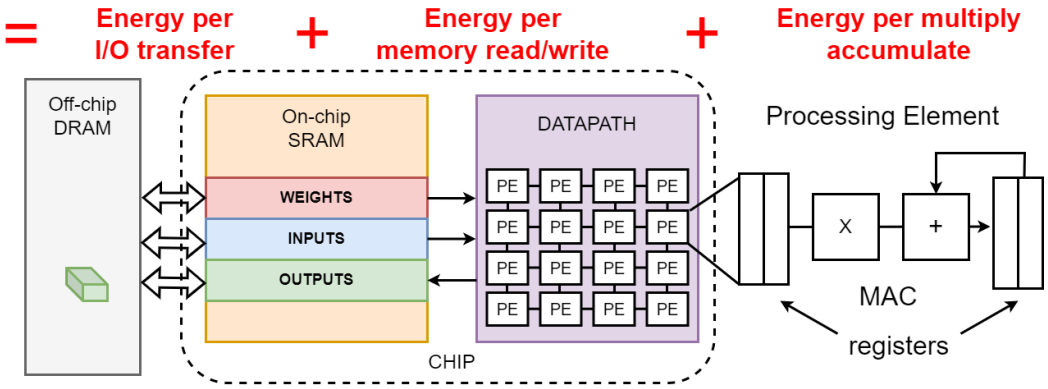
Energy Efficient DNN processors

ENERGY EFFICIENCY: Joule/op

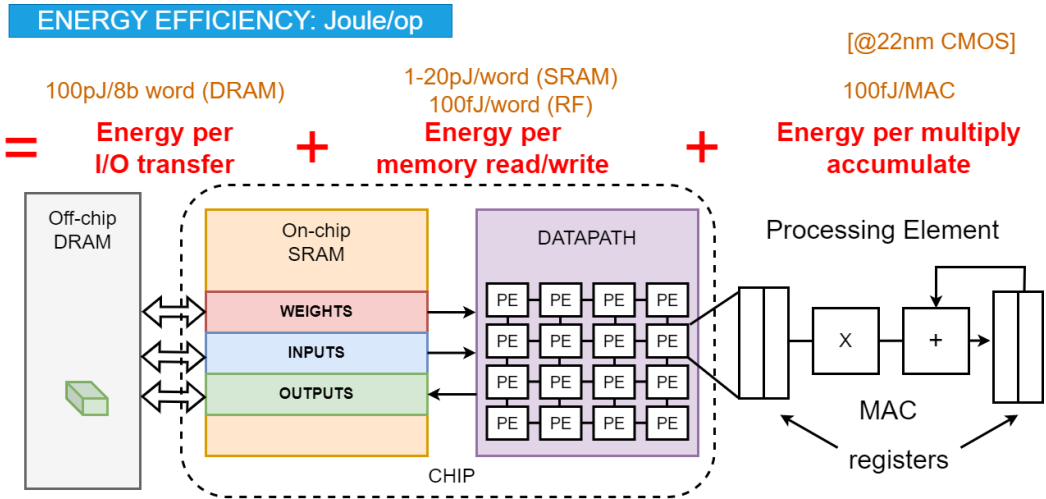


Energy Efficient DNN processors

ENERGY EFFICIENCY: Joule/op



Energy Efficient DNN processors



Energy Efficient DNN processors: challenges

Million of parameters in DNNs

- VGG-Net (2014), 28.25M
- GoogleNet (2015), 6.77M
- ResNet-50 (2016), 23M
- Megatron (2019), 8.3B
- Chat-GPT-3 (2022), 175B
- Chat-GPT-4 (2023), .. in the T (MoE model, not fully active)

Need parallel compute, bad for CPUs.

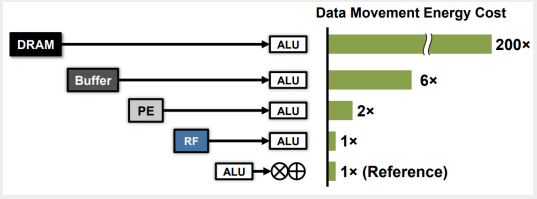
Energy Efficient DNN processors: challenges

Million of parameters in DNNs

- VGG-Net (2014), 28.25M
- GoogleNet (2015), 6.77M
- ResNet-50 (2016), 23M
- Megatron (2019), 8.3B
- Chat-GPT-3 (2022), 175B
- Chat-GPT-4 (2023), .. in the T (MoE model, not fully active)

Need parallel compute, bad for CPUs.

Energy cost of data movement



Need reduction of energy, bad for GPUs.
[V. Sze et al. 2017]

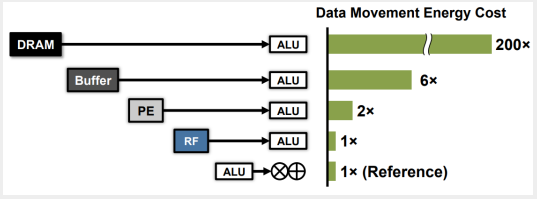
Energy Efficient DNN processors: challenges

Million of parameters in DNNs

- VGG-Net (2014), 28.25M
- GoogleNet (2015), 6.77M
- ResNet-50 (2016), 23M
- Megatron (2019), 8.3B
- Chat-GPT-3 (2022), 175B
- Chat-GPT-4 (2023), .. in the T (MoE model, not fully active)

Need parallel compute, bad for CPUs.

Energy cost of data movement



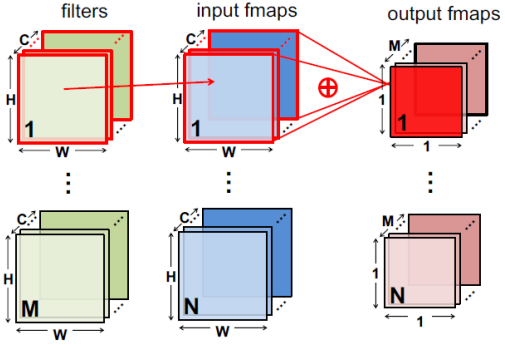
Need reduction of energy, bad for GPUs.
[V. Sze et al. 2017]

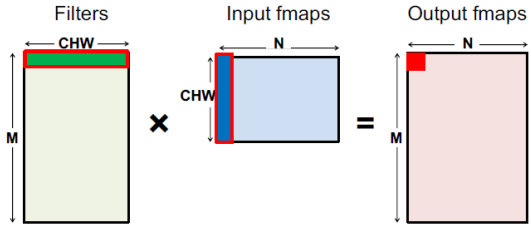
DNN accelerators

How DNN accelerators address these two fundamental challenges?

Map DNN to a Matrix Multiplication

Fully connected layer can be directly represented as matrix multiplication

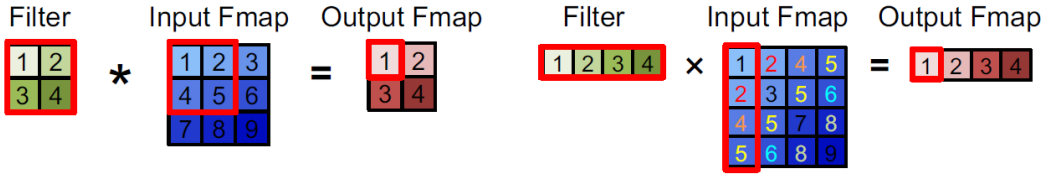




Note: Matrix multiplication also heavily used by recurrent and attention layers

Map DNN to a Matrix Multiplication

Convolutional layer can be converted to Toeplitz Matrix



Convolution

Matrix Multiply (by Toeplitz Matrix)

Data is repeated

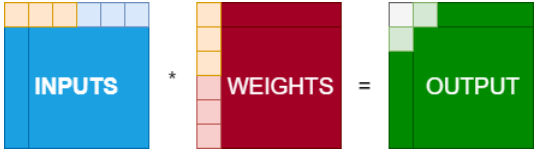
[Animation]

[Toepliz Matrix - Wiki]

Traditional GPUs and CPUs

If insufficient parallelism ⇒ partly spatial and partly temporal unrolling

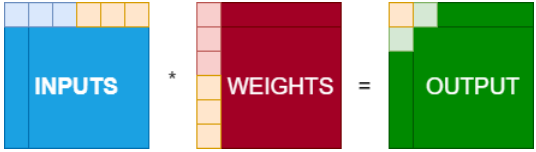
```
for (b=0 to B -1); for each image in the batch
        for (k=0 to K-1); for each output channel (temporal unrolling)
                for (c2=0 to C2/S-1); for each input channel (temporal unrolling)
                        parfor (c1=0 to S-1); for each input channel (spatial unrolling)
                                o[b][k] += i[b][c] * w[c][k]
```



Traditional GPUs and CPUs

If insufficient parallelism ⇒ partly spatial and partly temporal unrolling

```
for (b=0 to B -1); for each image in the batch
        for (k=0 to K-1); for each output channel (temporal unrolling)
                for (c2=0 to C2/S-1); for each input channel (temporal unrolling)
                        parfor (c1=0 to S-1); for each input channel (spatial unrolling)
                                o[b][k] += i[b][c] * w[c][k]
```



Traditional GPUs and CPUs

If insufficient parallelism: partly spatial and partly temporal unrolling

```
module convolution_unrolling #(parameter B = 8, K = 16, C2 = 32, S = 4) (
 input logic clk,
 input logic rst,
 input logic signed [15:0] i[B][C2], // Input tensor
 input logic signed [15:0] w[C2][K], // Weight tensor
 output logic signed [15:0] o[B][K]  // Output tensor
);
 always_ff @(posedge clk or posedge rst) begin
  if (rst) begin
   for (int b = 0; b < B; b++)
    for (int k = 0; k < K; k++)
     o[b][k] <= 0;
  end
  // continued in the next slide ..
```

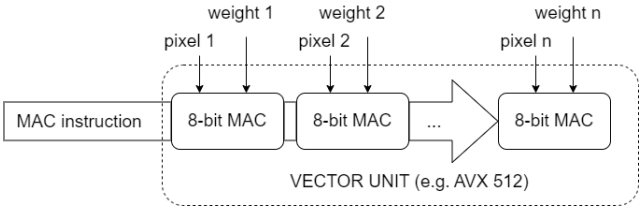
Traditional GPUs and CPUs

If insufficient parallelism: partly spatial and partly temporal unrolling

```
  // continued from previous slide..
  else begin
   for (int b = 0; b < B; b++) begin // For each image in the   batch
    for (int k = 0; k < K; k++) begin // For each output channel (temporal unro
     for (int c2 = 0; c2 < (C2/S); c2++) begin // Temporal unrolling
      for (int c1 = 0; c1 < S; c1++) begin // Spatial unrolling
       int c = c2 * S + c1;
       o[b][k] <= o[b][k] + i[b][c] * w[c][k];
      end
     end
    end
   end
  end
 end
endmodule
```

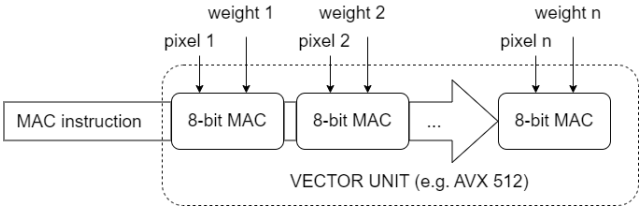
Deep Learning on CPUs

- Single Instruction Multiple Data (SIMD) function, parallel MAC in floating point, e.g., AVX 512 = 512-bit SIMD (e.g., 64 x 8-bit SIMD)
- Super scalar processor (parallel load, store, and compute)



Deep Learning on CPUs

- Single Instruction Multiple Data (SIMD) function, parallel MAC in floating point, e.g., AVX 512 = 512-bit SIMD (e.g., 64 x 8-bit SIMD)
- Super scalar processor (parallel load, store, and compute)

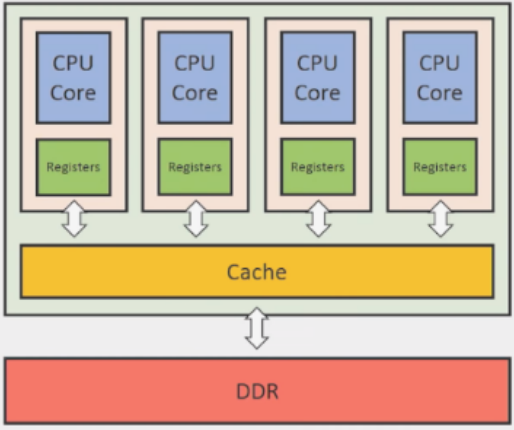


CPUs are not efficient for DNN

Limited parallelism, individual weights and fetch operations ($\sim 10 GOPS/W$) due to overhead of caching, decoding, but we need more!

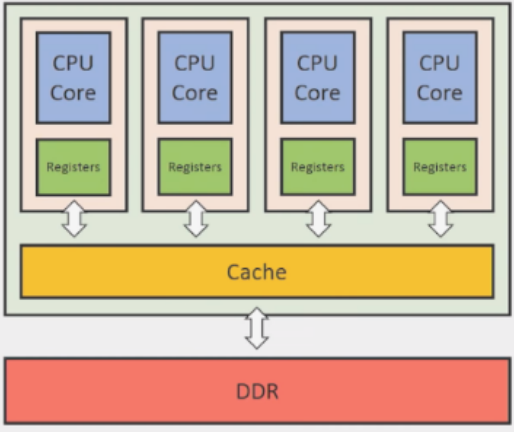
CPUs vs GPUs

Typical CPU architecture

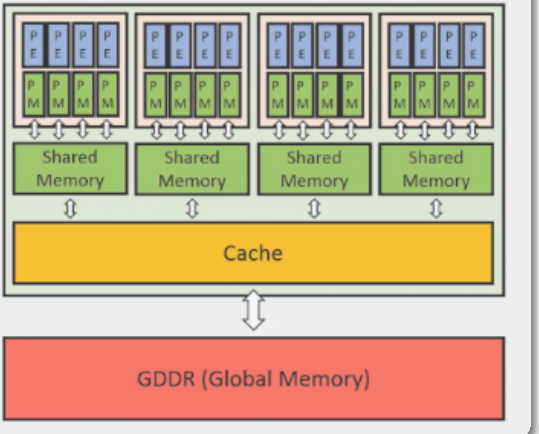


CPUs vs GPUs

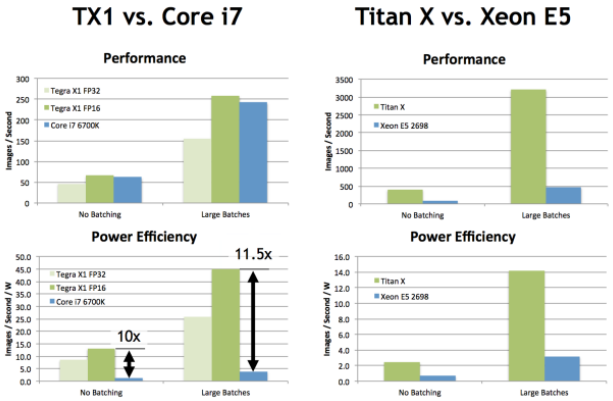
Typical CPU architecture



Typical GPU architecture



NN inference on GPUs



[source NVIDIA]

- Embedded GPUs achieve the same throughput as big CPUs
- Yet at 10-100x lower power (10-100 GOPS/W)
- still not enough performance!

CPUs vs GPUs

Typical CPU architecture

- **Execution**: small number of cores, separate instructions on each core independently
- **Memory**: low-bandwidth, random access
- **Power Consumption**: high-power
- Independent task parallelism

CPUs vs GPUs

Typical CPU architecture

- **Execution**: small number of cores, separate instructions on each core independently
- **Memory**: low-bandwidth, random access
- **Power Consumption**: high-power
- Independent task parallelism

Typical GPU architecture

- **Execution**: large number small execution units, single instructions on multiple cores
- **Memory**: hierarchical, predictable access, high-bandwidth
- **Power Consumption**: high-power (efficient for vector operations)
- Large data parallelism

CPUs vs GPUs

Typical CPU architecture

- **Execution**: small number of cores, separate instructions on each core independently
- **Memory**: low-bandwidth, random access
- **Power Consumption**: high-power
- Independent task parallelism

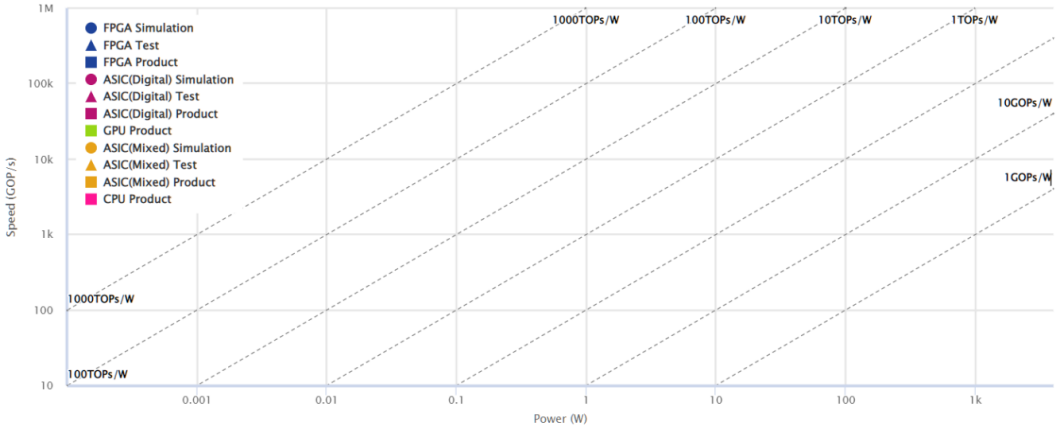
Typical GPU architecture

- **Execution**: large number small execution units, single instructions on multiple cores
- **Memory**: hierarchical, predictable access, high-bandwidth
- **Power Consumption**: high-power (efficient for vector operations)
- Large data parallelism

Training

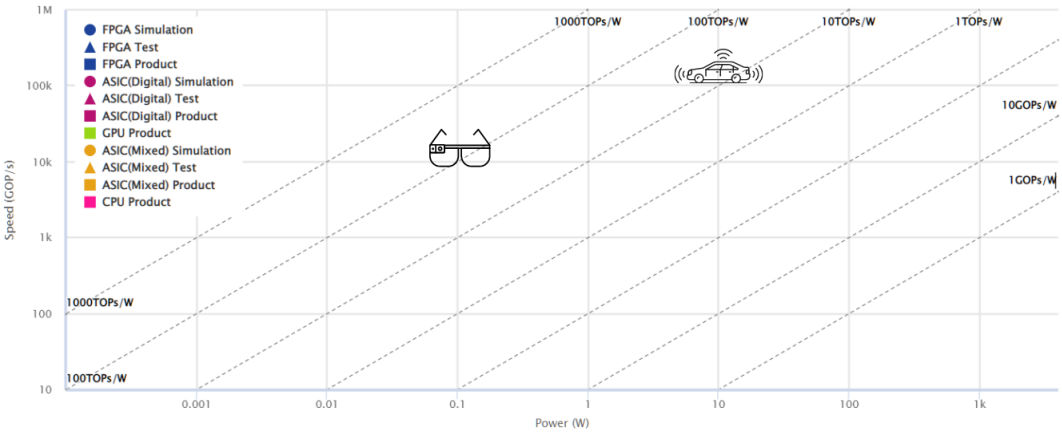
GPUs are optimal for batch-training, but suboptimal for low-energy inference.

Where are we?



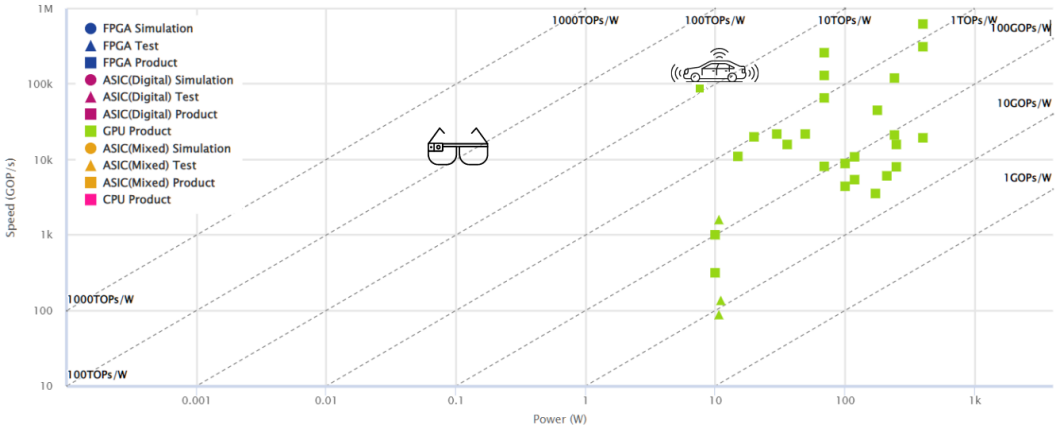
source nicsefc.ee.tsinghua.edu.cn

Where are we?



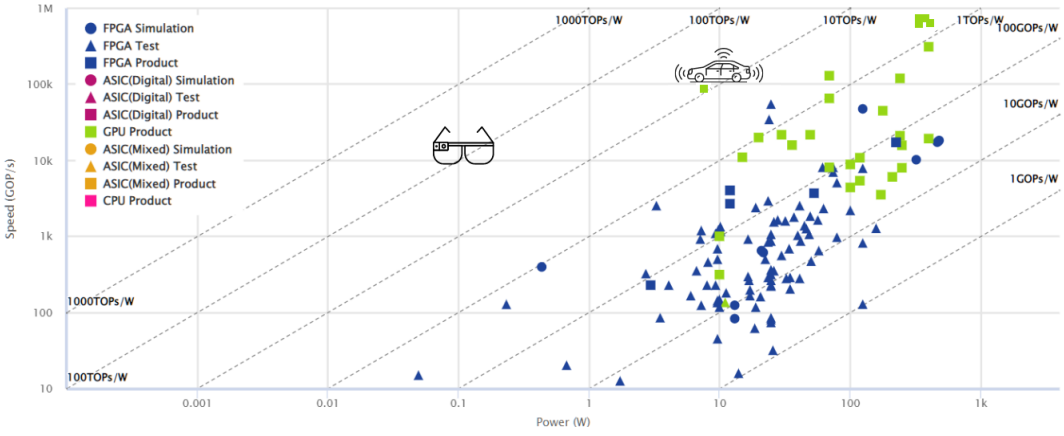
source nicsefc.ee.tsinghua.edu.cn

Where are we?



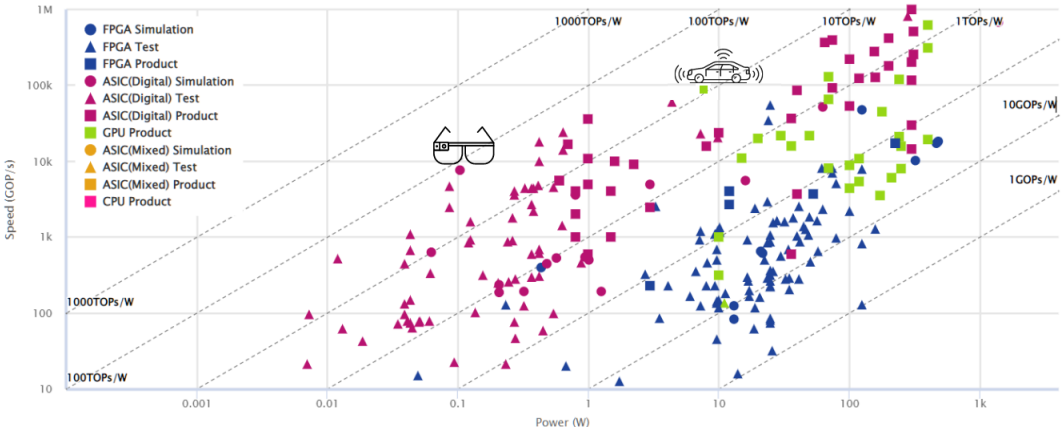
source nicsefc.ee.tsinghua.edu.cn

Where are we?



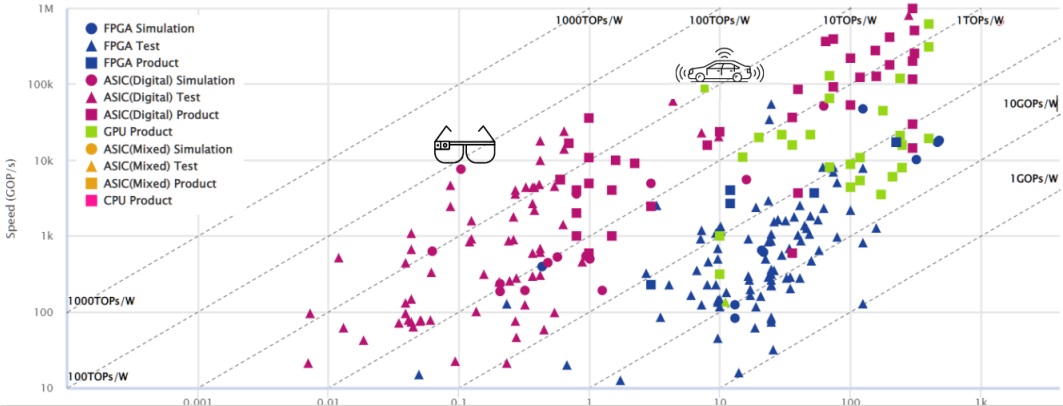
source nicsefc.ee.tsinghua.edu.cn

Where are we?



source nicsefc.ee.tsinghua.edu.cn

Where are we?

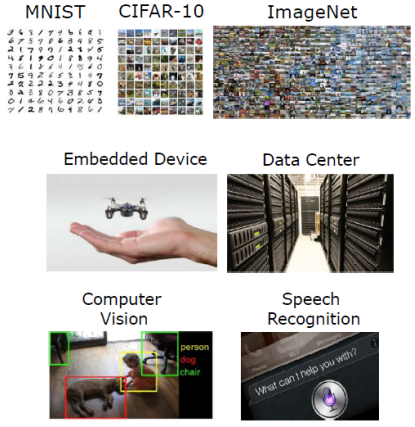


Are we there?

Unfortunately, not yet! These numbers report only peak efficiency and not workload efficiency... but we are getting close.

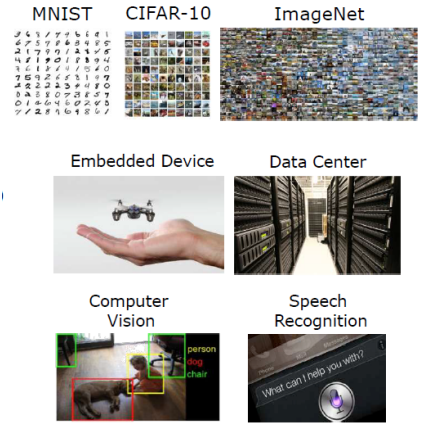
Outline

Key metrics and design objectives: much more than Ops/W!



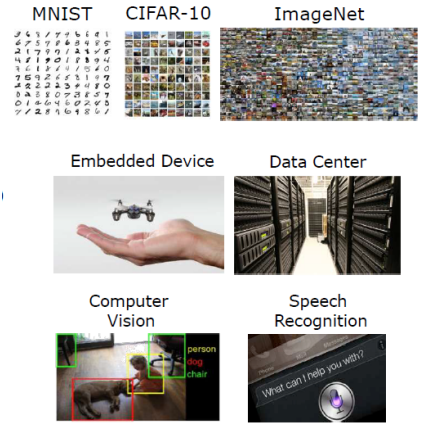
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)



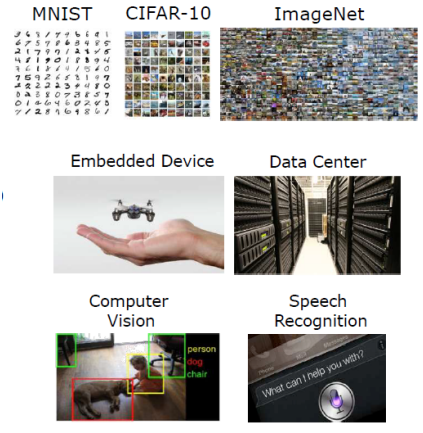
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)



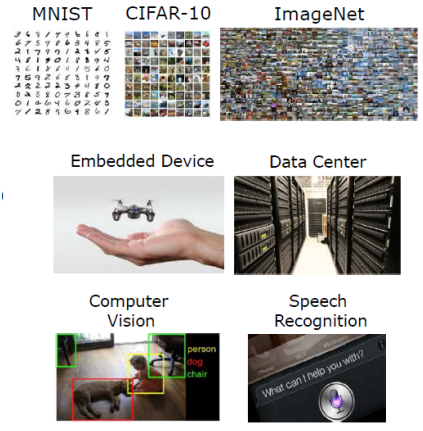
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)
3. Latency
   - For interactive applications (e.g., autonomous navigation)



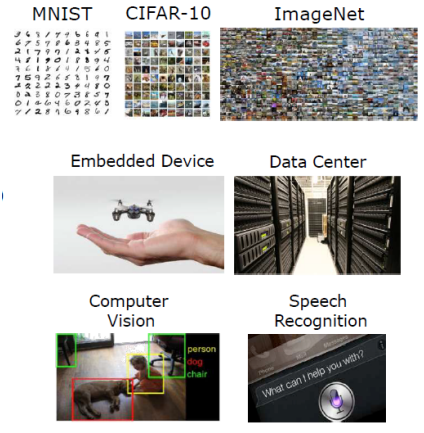
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)
3. Latency
   - For interactive applications (e.g., autonomous navigation)
4. Energy (embedded devices)
5. Power (data centers)



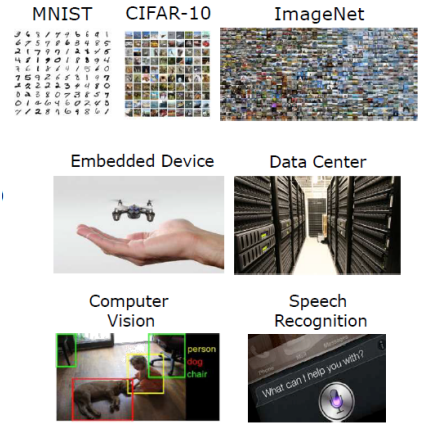
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)
3. Latency
   - For interactive applications (e.g., autonomous navigation)
4. Energy (embedded devices)
5. Power (data centers)
6. Hardware Cost (n $)
7. Flexibility



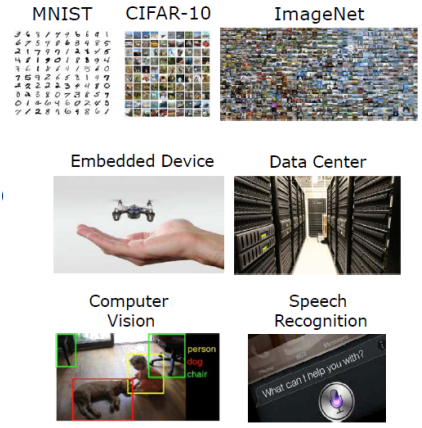
Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)
3. Latency
   - For interactive applications (e.g., autonomous navigation)
4. Energy (embedded devices)
5. Power (data centers)
6. Hardware Cost (n $)
7. Flexibility
   - Range of DNN models and tasks



Key metrics and design objectives: much more than Ops/W!

1. Accuracy (quality of result)
2. Throughput
   - Analytics on high volume data
   - Real-time performance (e.g., video at 30 fps)
3. Latency
   - For interactive applications (e.g., autonomous navigation)
4. Energy (embedded devices)
5. Power (data centers)
6. Hardware Cost (n $)
7. Flexibility
   - Range of DNN models and tasks
8. Scalability
   - Scaling of performance with amount of resources



Outline

Key metrics and design objectives: Accuracy

1. Accuracy

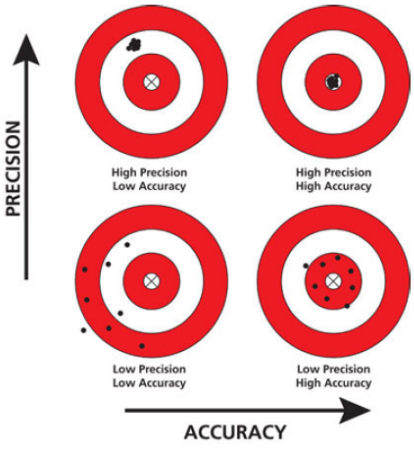
Accuracy is used to indicate the quality of the result for a give task. Units to measure accuracy depends on the task.

Key metrics and design objectives: Accuracy

- acc, mAP, ROC

1. Accuracy

Accuracy is used to indicate the quality of the result for a give task. Units to measure accuracy depends on the task.



Key metrics and design objectives: Throughput & Latency

2. Throughput

Throughput is used to indicate the amount of processed data. High-throughput is often critical to an application (e.g., video streaming).

3. Latency

Latency measures the time between the input data (ready) and the output result is generated. Low-latency is critical for real-time interactive applications. (e.g. Automatic Emergency Braking (AEB)).

Throughput $\neq$ Latency

For example, batching input data can increase throughput since it amortizes the overhead, as loading the weights; however, batching also increases latency, which can be unacceptable in certain applications.

Key metrics and design objectives: Throughput & Latency

Factors that affect throughput and latency:

$$\frac{inferences}{second} = \frac{operation}{second} x \frac{1}{\frac{operation}{inference}} \tag{2}$$

Key metrics and design objectives: Throughput & Latency

Factors that affect throughput and latency:

$$\frac{inferences}{second} = \frac{operation}{second} x \frac{1}{\frac{operation}{inference}} \tag{2}$$

$$\frac{operation}{second} = \frac{1}{\frac{cycle}{operation}} x \frac{cycle}{second} \tag{3}$$

Key metrics and design objectives: Throughput & Latency

Factors that affect throughput and latency:

$$\frac{inferences}{second} = \frac{operation}{second} x \frac{1}{\frac{operation}{inference}} \tag{2}$$

$$\frac{operation}{second} = \frac{1}{\frac{cycle}{operation}} x \frac{cycle}{second} \tag{3}$$

and, for a system with multiple processing elements (PE)? How should we adapt eq 3?

Key metrics and design objectives: Throughput & Latency

$$\frac{operation}{second} = \frac{1}{\frac{cycle}{operation}} x \frac{cycle}{second} x \quad number \quad of \quad PEs \quad x \quad utilization \quad of \quad PEs \qquad (4)$$

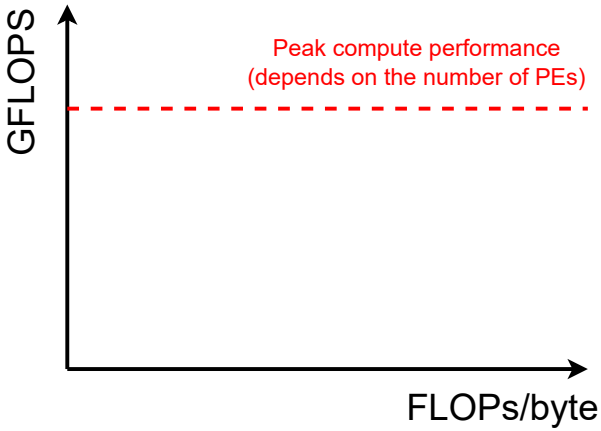
Key metrics and design objectives: Throughput & Latency

$$\frac{operation}{second} = \frac{1}{\frac{cycle}{operation}} x \frac{cycle}{second} x \quad number \quad of \quad PEs \quad x \quad utilization \quad of \quad PEs \qquad (4)$$

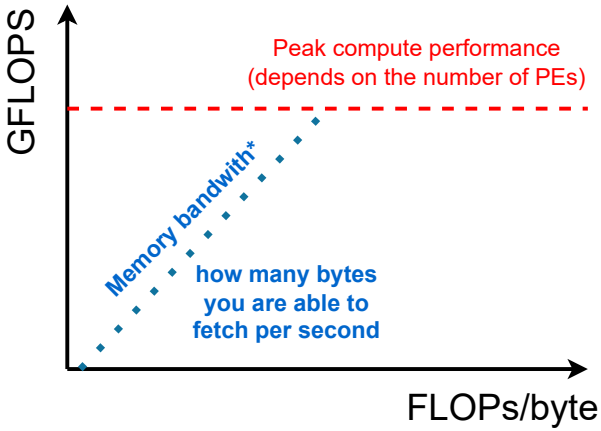
$$utilization \quad of \quad PEs = \frac{number \quad of \quad active \quad PE}{number \quad of \quad PEs} \quad x \quad utilization \quad of \quad active \quad PEs \qquad (5)$$

Key metrics and design objectives: the roofline model

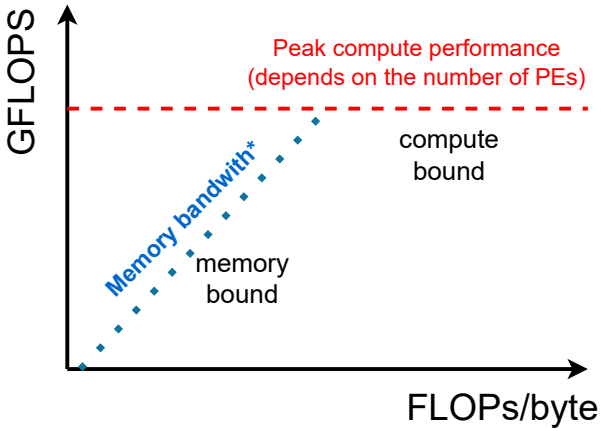
Key metrics and design objectives: the roofline model



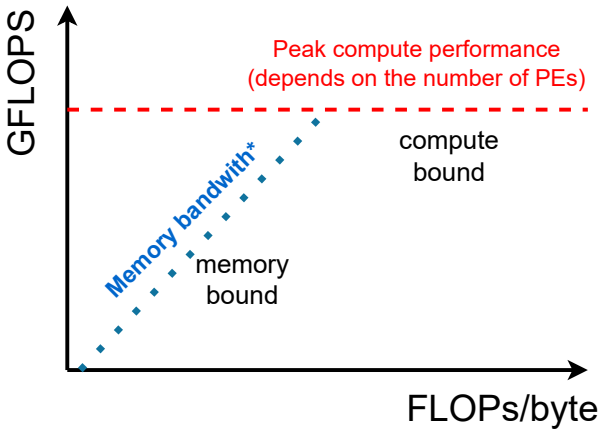
Key metrics and design objectives: the roofline model



Key metrics and design objectives: the roofline model



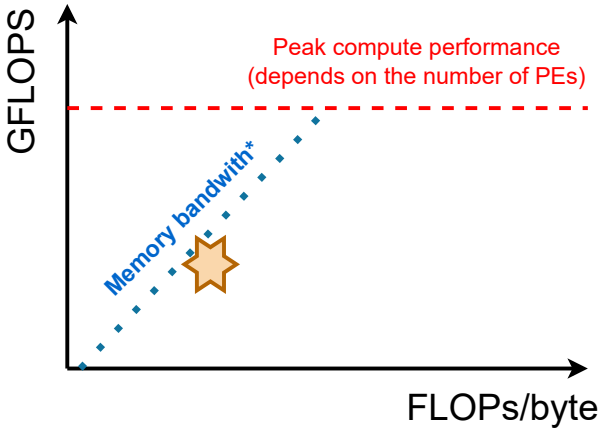
Key metrics and design objectives: the roofline model



Suppose:

- 100 PEs at 1GHz $\rightarrow$ 100 GFLOPS
- each PE needs 2 bytes of read and 1 bytes of write (e.g., 8-bit weights and output)
- DRAM BW $\sim$ 25GBps

Key metrics and design objectives: the roofline model



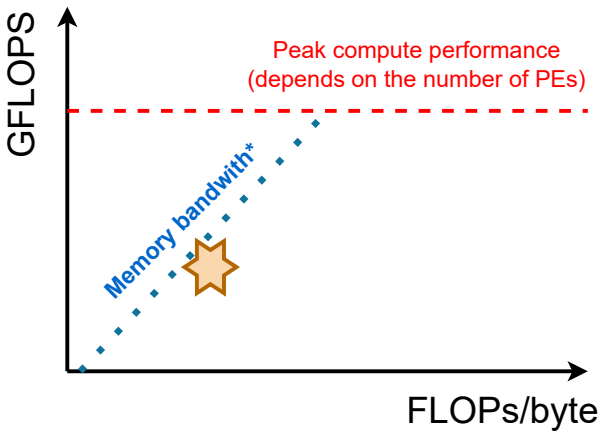
Suppose:

- 100 PEs at 1GHz $\rightarrow$ 100 GFLOPS
- each PE needs 2 bytes of read and 1 bytes of write (e.g., 8-bit weights and output)
- DRAM BW $\sim$ 25GBps

Memory or compute bound?

- BW requirements 3 bytes/cycle $\rightarrow$ 300 GBbs
- We are memory bound!

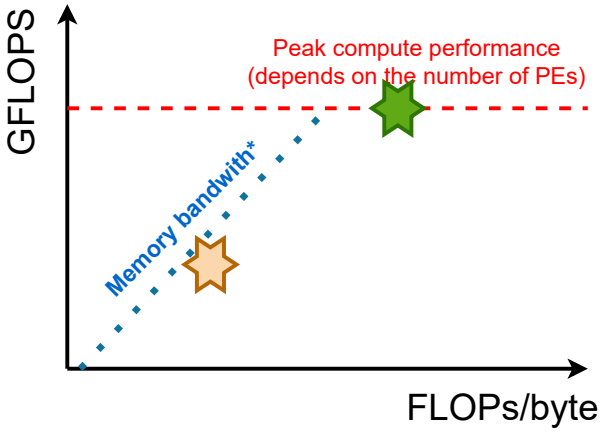
Key metrics and design objectives: the roofline model



Suppose:

- 100 PEs at 1GHz $\rightarrow$ 100 GFLOPS
- each PE needs 2 bytes of read and 1 bytes of write
- DRAM BW $\sim$ 25GBps
- but we reuse the data!
  - weights reused completely
  - input reused 10 times
  - psum reused 10 times

Key metrics and design objectives: the roofline model



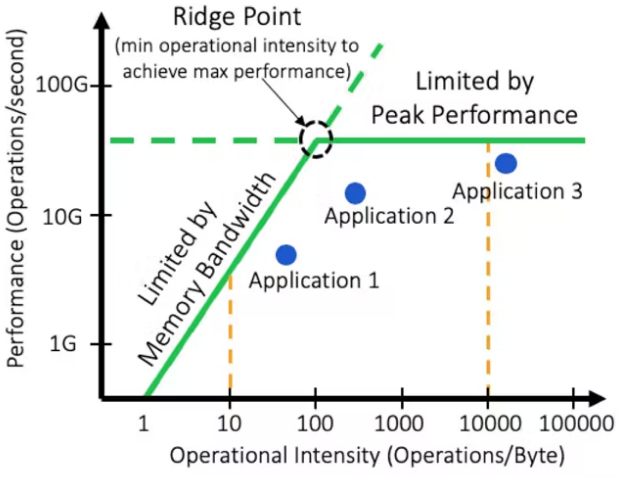
Suppose:

- 100 PEs at 1GHz $\rightarrow$ 100 GFLOPS
- each PE needs 2 bytes of read and 1 bytes of write
- DRAM BW $\sim$ 25GBps
- but we reuse the data!
  - weights reused completely
  - input reused 10 times
  - psum reused 10 times

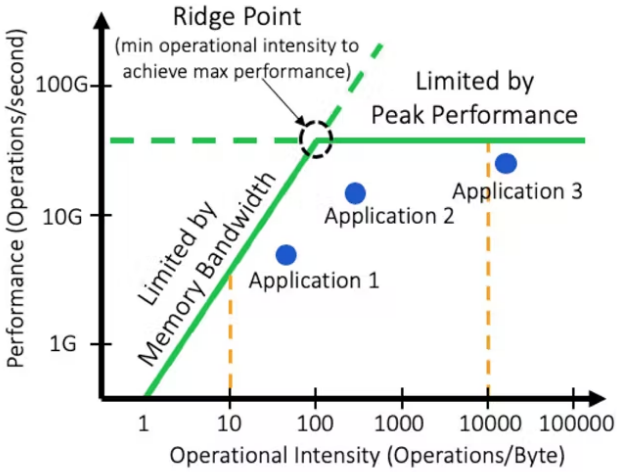
Memory or compute bound?

- BW 2 bytes/10 cycle $\rightarrow$ 20 GBps
- We are compute bound!

Key metrics and design objectives: the roofline model



Key metrics and design objectives: the roofline model



The roofline model

When the operational intensity, which dictates the amount of compute per byte of data, is low, the operation per second is limited by the data delivery. The design goal is to operate as close as possible to the peak operation per second for a given workload.

[Wikipedia - the roofline model]

Key metrics and design objectives: Throughput & Latency

Increase Throughput and Reduce Latency

- Reduce time per MAC
  - Reduce critical path $\Rightarrow$ increase clock frequency
  - Reduce instruction overhead

Key metrics and design objectives: Throughput & Latency

Increase Throughput and Reduce Latency

- Reduce time per MAC
  - Reduce critical path $\Rightarrow$ increase clock frequency
  - Reduce instruction overhead
- Avoid unnecessary MACs (save cycles by e.g., skipping multiplication with zeros)

Key metrics and design objectives: Throughput & Latency

Increase Throughput and Reduce Latency

- Reduce time per MAC
  - Reduce critical path $\Rightarrow$ increase clock frequency
  - Reduce instruction overhead
- Avoid unnecessary MACs (save cycles by e.g., skipping multiplication with zeros)
- Increase number of processing elements (PE) $\Rightarrow$ more MACs in parallel
  - Increase area density of PE or area cost of system

Key metrics and design objectives: Throughput & Latency

Increase Throughput and Reduce Latency

- Reduce time per MAC
  - Reduce critical path $\Rightarrow$ increase clock frequency
  - Reduce instruction overhead
- Avoid unnecessary MACs (save cycles by e.g., skipping multiplication with zeros)
- Increase number of processing elements (PE) $\Rightarrow$ more MACs in parallel
  - Increase area density of PE or area cost of system
- Increase PE utilization $\Rightarrow$ keep PEs busy
  - Distribute workload to as many PEs as possible
  - Balance the workload across PEs
  - Sufficient memory bandwidth to deliver workload to PEs (reduce idle cycles)

Key metrics and design objectives: Throughput & Latency

Increase Throughput and Reduce Latency

- Reduce time per MAC
  - Reduce critical path $\Rightarrow$ increase clock frequency
  - Reduce instruction overhead
- Avoid unnecessary MACs (save cycles by e.g., skipping multiplication with zeros)
- Increase number of processing elements (PE) $\Rightarrow$ more MACs in parallel
  - Increase area density of PE or area cost of system
- Increase PE utilization $\Rightarrow$ keep PEs busy
  - Distribute workload to as many PEs as possible
  - Balance the workload across PEs
  - Sufficient memory bandwidth to deliver workload to PEs (reduce idle cycles)
- Low latency has an additional constraint of small batch size

Key metrics and design objectives: Energy Efficiency & Power Cons.

4. Energy Efficiency

Energy efficiency indicates the amount of data that can be processed or the number of executions of a task that can be completed for a given unit of energy. High energy efficiency is important for all applications and is reported as number of operation per joule.

Key metrics and design objectives: Energy Efficiency & Power Cons.

4. Energy Efficiency

Energy efficiency indicates the amount of data that can be processed or the number of executions of a task that can be completed for a given unit of energy. High energy efficiency is important for all applications and is reported as number of operation per joule.

5. Power Consumption

Power Consumption is the amount of energy consumed per unit of time. More power cons. $\Rightarrow$ increase heat dissipation. At the design time, thermal design power (TDP) is the power the cooling system is designed to dissipate.

Key metrics and design objectives: Energy Efficiency & Power Cons.

Limits

Power consumption and energy efficiency limits the throughput as follows:

$$\frac{inferences}{second} \leq Max\left(\frac{Joules}{second}\right) x \frac{inferences}{joules} \tag{6}$$

Key metrics and design objectives: Energy Efficiency & Power Cons.

Limits

Power consumption and energy efficiency limits the throughput as follows:

$$\frac{inferences}{second} \leq Max\left(\frac{Joules}{second}\right) x \frac{inferences}{joules} \tag{6}$$

the number of inference per joule can be decomposed into

$$\frac{inferences}{joule} = \frac{operationgs}{joule} x \frac{1}{\frac{operatins}{inference}} \tag{7}$$

where the number of operations per joule depend on both the hardware and the DNN model, while the number of operations per inference is dictated by the DNN model.

Key metrics and design objectives: Energy Efficiency & Power Cons.

Design considerations for the HW affecting energy per operation

- data movement
- energy per MAC

$$Energy_{total} = Energy_{data} + Energy_{MAC} \quad (8)$$

$$\frac{joules}{operation} = \alpha x C x V_{DD}^2 \quad (9)$$

C = total switching capacitance
α = switching activity
V_{DD} = supply voltage

Key metrics and design objectives: Energy Efficiency & Power Cons.

Design considerations for the HW affecting energy per operation

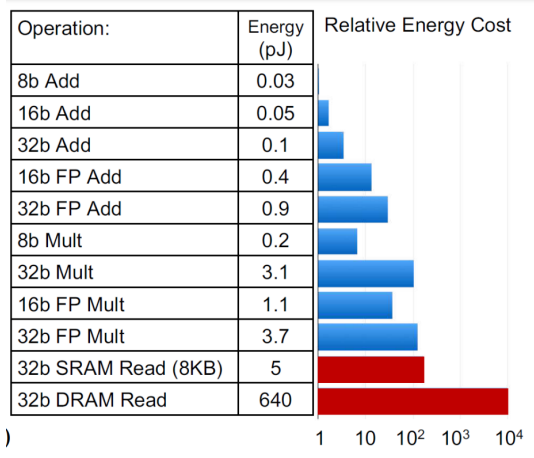
- data movement
- energy per MAC

$$Energy_{total} = Energy_{data} + Energy_{MAC} \quad (8)$$

$$\frac{joules}{operation} = \alpha x C x V_{DD}^2 \quad (9)$$

C = total switching capacitance
α = switching activity
V_{DD} = supply voltage

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



Horowitz, 2014

Key metrics and design objectives: Energy Efficiency & Power Cons.

Remarks and considerations

- Off-chip memory access under control (e.g. memory hierarchy)
- micro-architecture optimizations for low-switching activity (i.e., small α by reducing precision)
- reducing instruction bookkeeping (e.g., single-instruction, multiple-data (SIMD), single-instruction multiple-threads (SIMT)).
- exploit sparsity at the HW & SW level (e.g. prune, quantization)

Key metrics and design objectives: Hardware Cost

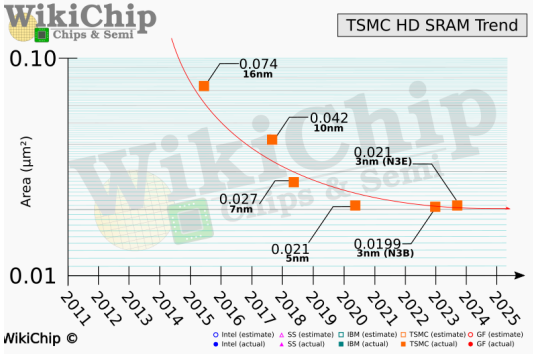
6. Hardware Cost

Chip area in conjunction with process technology node constrains the amount of chip-storage and compute (e.g. number of PEs, on-chip SRAM). Off-chip bandwidth dictates the costs of complexity of the packaging and PCB.

Key metrics and design objectives: Hardware Cost

6. Hardware Cost

Chip area in conjunction with process technology node constrains the amount of chip-storage and compute (e.g. number of PEs, on-chip SRAM). Off-chip bandwidth dictates the costs of complexity of the packaging and PCB.



WikiChip IEDM 2022

Key metrics and design objectives: Flexibility

7. Flexibility

Flexibility refers to the range of DNN models supported by the processor and the ability of the SW environment (e.g., mapper, compilers).

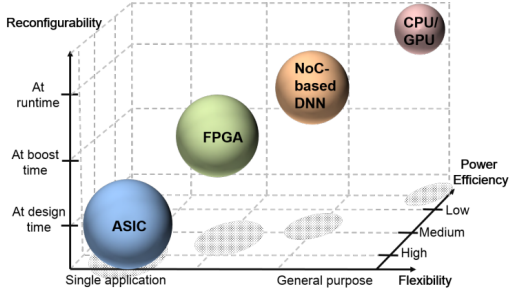
Flexibility vs Efficiency trade-off

Key metrics and design objectives: Flexibility

7. Flexibility

Flexibility refers to the range of DNN models supported by the processor and the ability of the SW environment (e.g., mapper, compilers).

Flexibility vs Efficiency trade-off



Kun-Chih (Jimmy) Chen et. al (ACM) 2019

Key metrics and design objectives: Scalability

8. Scalability

Scalability refers to scaling up model size (DNN) and also building systems with multiple chips (chiplets), or even wafer-scale chips. Sometimes, scalability refers to how well a design can be scaled up to achieve higher throughput and energy efficiency when increasing the amount of resources.

Key metrics and design objectives: Summary

1. Accuracy
2. Throughput
3. Latency
4. Energy Efficiency
5. Power Consumption
6. Hardware Cost
7. Flexibility
8. Scalability

Complex Optimization Problem

The design of HW accelerator is a complex business. The metrics of interest have several dependencies and a good evaluation process is necessary to determine the soundness of a good DNN HW solution.

Key metrics and design objectives: Summary

Comprehensive Coverage for Evaluation

- All metrics should be reported for fair evaluation of design tradeoffs
- Examples of what can happen if a certain metric is omitted:

Key metrics and design objectives: Summary

Comprehensive Coverage for Evaluation

- All metrics should be reported for fair evaluation of design tradeoffs
- Examples of what can happen if a certain metric is omitted:
  - Without the accuracy given for a specific dataset and task, one could run a simple DNN and claim low power, high throughput, and low cost, however, the processor might not be usable for a meaningful task
  - Without reporting the off-chip memory access, one could build a processor with only MACs and claim low cost, high throughput, high accuracy, and low chip power . However, when evaluating system power, the off-chip memory access would be substantial

Key metrics and design objectives: Summary

Comprehensive Coverage for Evaluation

- All metrics should be reported for fair evaluation of design tradeoffs
- Examples of what can happen if a certain metric is omitted:
  - Without the accuracy given for a specific dataset and task, one could run a simple DNN and claim low power, high throughput, and low cost, however, the processor might not be usable for a meaningful task
  - Without reporting the off-chip memory access, one could build a processor with only MACs and claim low cost, high throughput, high accuracy, and low chip power . However, when evaluating system power, the off-chip memory access would be substantial
- Are results measured or simulated? On what test data?

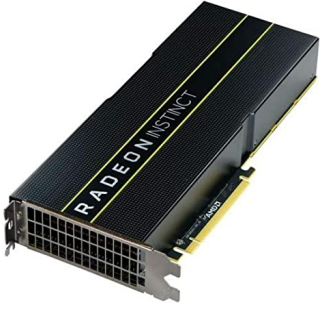
Outline

CPU vs GPUs

Intel Xeon (Cascade Lake)

Nvidia Tesla (Volta)

AMD Radeon (Instinct)



Matrix Multiplication

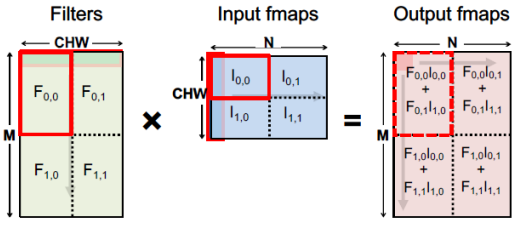
Use matrix multiplication libraries on CPUs and GPUs

CPU, GPU Libraries for Matrix Multiplication

- Implementation: Matrix Multiplication (GEMM)
  - CPU: OpenBLAS, Intel MKL, etc
  - GPU: cuBLAS, cuDNN, etc
- Library will note shape of the matrix multiply and select an implementation optimized for that shape
- Optimization usually involves proper tiling to memory hierarchy

Tiling Matrix Multiplication

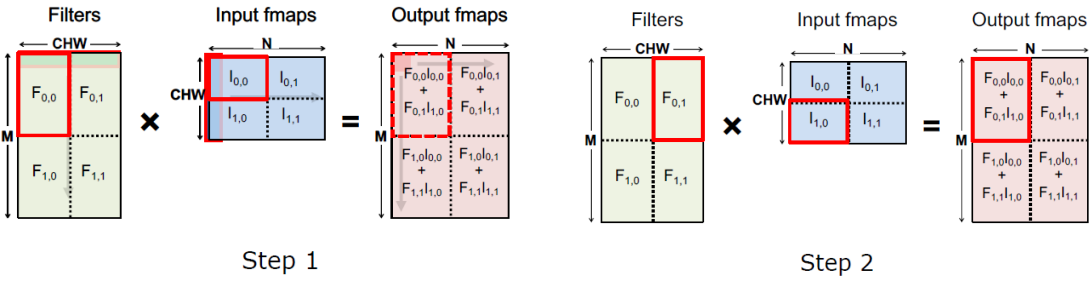
Matrix multiplication tiled to fit in cache (i.e., on-chip memory) and computation ordered to maximize reuse of data in cache



Step 1

Tiling Matrix Multiplication

Matrix multiplication tiled to fit in cache (i.e., on-chip memory) and computation ordered to maximize reuse of data in cache



Analogy: Gauss's Multiplication Algorithm

$$(a+bi)(c+di) = (ac-bd)(bc+ad)i \tag{10}$$

4 multiplications + 3 additions

Analogy: Gauss's Multiplication Algorithm

$$(a+bi)(c+di) = (ac-bd)(bc+ad)i \tag{10}$$

4 multiplications + 3 additions

$$k_1 = c\cdot(a+b) k_2 = a\cdot(d-c) k_3 = b\cdot(c+d) \tag{11}$$

3 multiplications + 5 additions

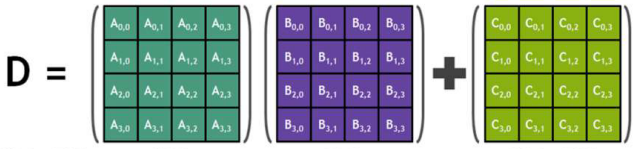
Real Part $= k_1 - k_3$
Imaginary Part $= k_1 + k_2$

Why?

Reduce number of multiplications to increase throughput

Reduce Instruction Overhead

- Perform more MACs per instruction
  - CPU: SIMD / Vector Instructions (e.g., Specialized Vector Neural Network Instructions (VNNI) fuse separate multiply and add instructions into single MAC instruction and avoid storing intermediate values in memory)
  - GPU: SIMT / Tensor Instructions (e.g., New opcode Matrix Multiply Accumulate (HMMA) performs 64 MACs with Tensor Core)
- Perform more MACs per cycle without increasing memory bandwidth by adding support for reduced precision
  - e.g., If access 512 bits per cycle, can perform 64 8-bit MACs vs. 16 32-bit MACs



Tensor Core Image Source: Nvidia

Design Considerations for CPU and GPU

- Software (compiler)
  - Reduce unnecessary MACs: Apply transforms
  - Increase PE utilization: Schedule loop order and tile data to increase data reuse in memory hierarchy
- Hardware
  - Reduce time per MAC
    - Increase speed of PEs
    - Increase MACs per instruction using large aggregate instructions (e.g., SIMD, tensor core) $\Rightarrow$ requires additional hardware
  - Increase number of parallel MACs
    - Increase number of PEs on chip $\Rightarrow$ area cost
    - Support reduced precision in PEs
  - Increase PE utilization
    - Increase on-chip storage $\Rightarrow$ area cost
    - External memory BW $\Rightarrow$ system cost

Outline

Summary & Quiz

Today

- The need of DNN accelerators
- DNN accelerator reference architecture
- CPU vs GPU
- Key metrics and design objectives (to read at home)
- The roofline model (to read at home)

Summary & Quiz

Today

- The need of DNN accelerators
- DNN accelerator reference architecture
- CPU vs GPU
- Key metrics and design objectives (to read at home)
- The roofline model (to read at home)

Next Steps

- GeMM & CNN processors enhancements in ASIC, CPU, GPU
  - Parallelization (spatial unrolling optimization)
  - Stationarity (temporal unrolling optimization)
  - Extending spatial and temporal unrolling to higher levels
- Notable Accelerator examples