Energy Efficiency and AI Hardware

Stanford AHA Retreat

August 31, 2023

Bill Dally

Chief Scientist and SVP of Research, NVIDIA Corporation

Adjunct Professor of CS and EE, Stanford

Q: How can AI hardware be made more energy efficient?

Improving the energy efficiency of AI hardware is a crucial goal to enable sustainable and scalable deployment of AI technologies. Here are several strategies and techniques that can be employed to make AI hardware more energy efficient:

1. **Specialized Hardware Design**:
    1. **ASICs (Application-Specific Integrated Circuits)**: Designing hardware specifically tailored to AI workloads can significantly improve energy efficiency. ASICs can be optimized for specific neural network operations, reducing unnecessary overhead.
    2. **FPGAs (Field-Programmable Gate Arrays)**: These reconfigurable hardware platforms can be programmed to accelerate specific AI tasks, providing better energy efficiency compared to general-purpose processors.
2. **Quantization and Pruning**:
    1. **Quantization**: Reducing the precision of numerical values used in computations (e.g., using 8-bit instead of 32-bit floating-point numbers) can lead to energy savings without a significant drop in accuracy.
    2. **Weight Pruning**: Removing or setting small weights to zero in neural networks reduces the computational workload and memory requirements, leading to energy savings during both training and inference.
3. **Low-Power Components**:
    1. **Low-Power Processors**: Using processors with low power consumption characteristics, such as mobile processors or those designed for IoT devices, can help reduce energy usage.
    2. **Efficient Memory Hierarchies**: Optimizing the memory subsystem to minimize data movement and maximize data reuse can lead to significant energy savings.
4. **Hardware-Software Co-Design**:
    1. **Model Optimization**: Collaborative efforts between hardware and software engineers can lead to model architectures that are more energy-efficient on a specific hardware platform.
    2. **Compiler Optimization**: Tailoring compilers to generate code that leverages hardware features efficiently can reduce energy overhead.
5. **Dynamic Voltage and Frequency Scaling (DVFS)**:
    1. **Dynamic Scaling**: Adapting the operating voltage and clock frequency of the hardware components based on workload requirements can lead to energy savings when the hardware is not fully utilized.
6. **Sleep States and Power Gating**:
    1. **Low-Power States**: Putting hardware components or entire sections into low-power sleep states when they're not actively needed can save energy.
    2. **Power Gating**: Completely shutting down certain hardware components when they're not needed can minimize leakage power consumption.
7. **Cooling and Thermal Management**:
    1. **Efficient Cooling**: Proper cooling solutions can prevent hardware components from overheating, which can cause energy inefficiencies and reduced lifespan.
8. **Energy-Aware Training**:
    1. **Regularization Techniques**: Methods like dropout and weight decay during training can lead to models that generalize better and require less computational resources for inference.
9. **Energy-Efficient Algorithms**:
    1. **Algorithmic Changes**: Exploring and using algorithms that require fewer computational steps or operations can contribute to energy savings.
10. **Hardware Acceleration**:

- **AI Hardware Accelerators**: Utilizing dedicated hardware accelerators designed for AI tasks, like GPUs or TPUs, can offer significant energy efficiency improvements.

11. **Dynamic Workload Management**:

- **Load Balancing**: Distributing workloads across multiple hardware components efficiently can prevent overloading some components while underutilizing others.

12. **Energy Monitoring and Optimization**:

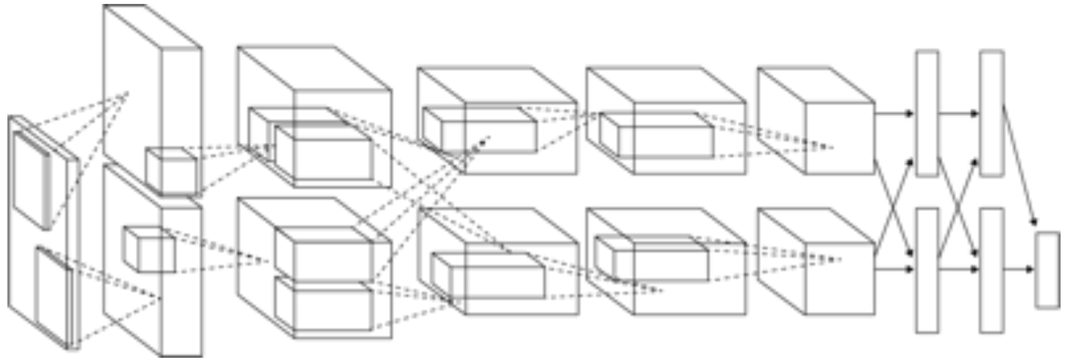
- **Real-Time Monitoring**: Implementing systems that continuously monitor energy consumption can enable dynamic optimization based on the workload and energy constraints.
- Overall, achieving energy efficiency in AI hardware requires a holistic approach involving hardware design, software optimization, algorithmic changes, and system-level considerations. Collaboration between hardware engineers, software developers, and AI researchers is essential to create energy-efficient AI solutions that align with the goals of sustainability and performance.

As of my last knowledge update in September 2021, Bill Dally is a prominent computer scientist and electrical engineer known for his work in computer architecture, particularly in the field of parallel processing and hardware acceleration. He has been involved in various projects and research related to energy-efficient AI hardware. While I can't provide direct quotes from him, I can offer insights into some of the principles and strategies he might advocate for making AI hardware more energy-efficient based on his previous work and statements:

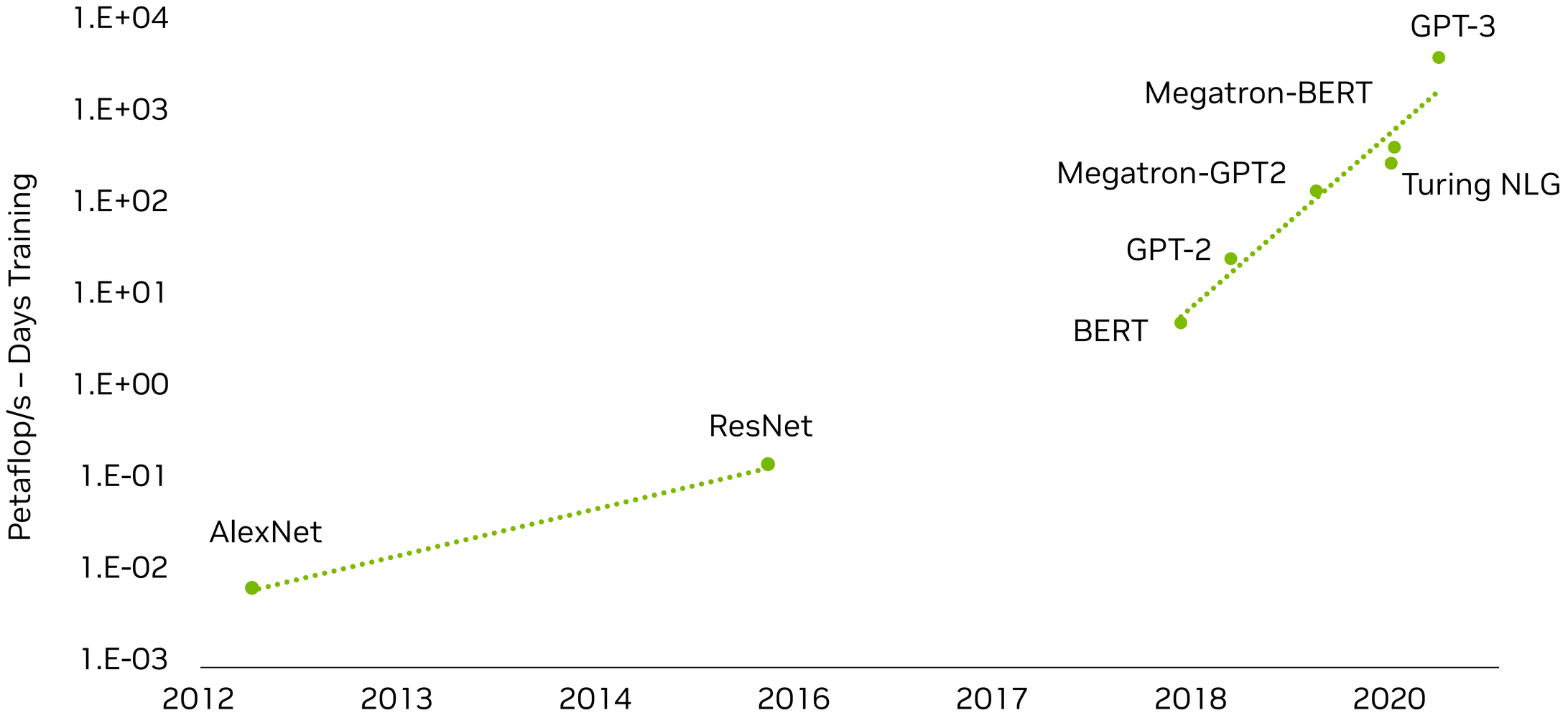
1. **Architectural Innovation**: Dally might emphasize the importance of rethinking the fundamental architecture of AI hardware to optimize for energy efficiency. He could propose exploring novel architectures that are specifically designed for AI workloads, leveraging concepts such as dataflow and specialized accelerators.
2. **Reduced Precision Computing**: Dally could advocate for the use of reduced-precision arithmetic (such as 8-bit or even lower) for AI computations. This approach can significantly reduce energy consumption while maintaining acceptable levels of accuracy in neural network computations.
3. **Hardware-Software Co-Design**: Dally might stress the significance of collaboration between hardware and software teams. He could suggest that close cooperation between these two groups is crucial for tailoring hardware designs to efficiently execute the types of computations commonly found in AI workloads.
4. **Sparsity and Pruning**: Dally could propose the integration of techniques like weight pruning and sparsity in neural networks. By eliminating unnecessary parameters and operations, AI hardware can be used more efficiently, resulting in reduced energy consumption.
5. **Memory Hierarchy Optimization**: Given Dally's expertise in parallel processing and memory systems, he might emphasize designing memory hierarchies that are optimized for AI workloads. Efficient memory access can lead to reduced energy overhead.
6. **Hybrid Computing**: Dally could suggest exploring hybrid computing approaches that combine general-purpose processors with specialized AI accelerators. This could involve offloading specific AI tasks to dedicated hardware components, thereby improving energy efficiency.
7. **Dynamic Voltage and Frequency Scaling (DVFS)**: Given his background, Dally might discuss the importance of dynamic voltage and frequency scaling to adapt hardware performance based on the current workload, leading to energy savings.
8. **Efficient Communication**: Dally could highlight the significance of minimizing data movement between different components of the hardware. Efficient communication mechanisms can prevent energy waste due to unnecessary data transfers.
9. **Energy-Efficient Interconnects**: Dally could propose using energy-efficient interconnect architectures to reduce the energy consumed in data transmission within the hardware.
10. **Power Management Techniques**: Dally might discuss power gating, where inactive components are temporarily powered off to save energy, as well as the importance of fine-tuned power management strategies.

Please note that these are speculative insights based on Bill Dally's previous work and contributions to the field of computer architecture. For the most up-to-date and accurate information, it's recommended to refer to his recent publications, talks, or interviews.

Deep Learning was Enabled by Hardware



Deep Learning is Gated by Hardware



$$E = \frac{1}{2}CV^2$$

$$E = \frac{1}{2}CV^2$$

V: Reduce voltage - to the point it starts getting too slow

- ~0.5V today
- 2x vs 0.7v, 4x vs 1.0v
- Too slow when leakage or cost excessive

$$E = \frac{1}{2}CV^2$$

C: Three components

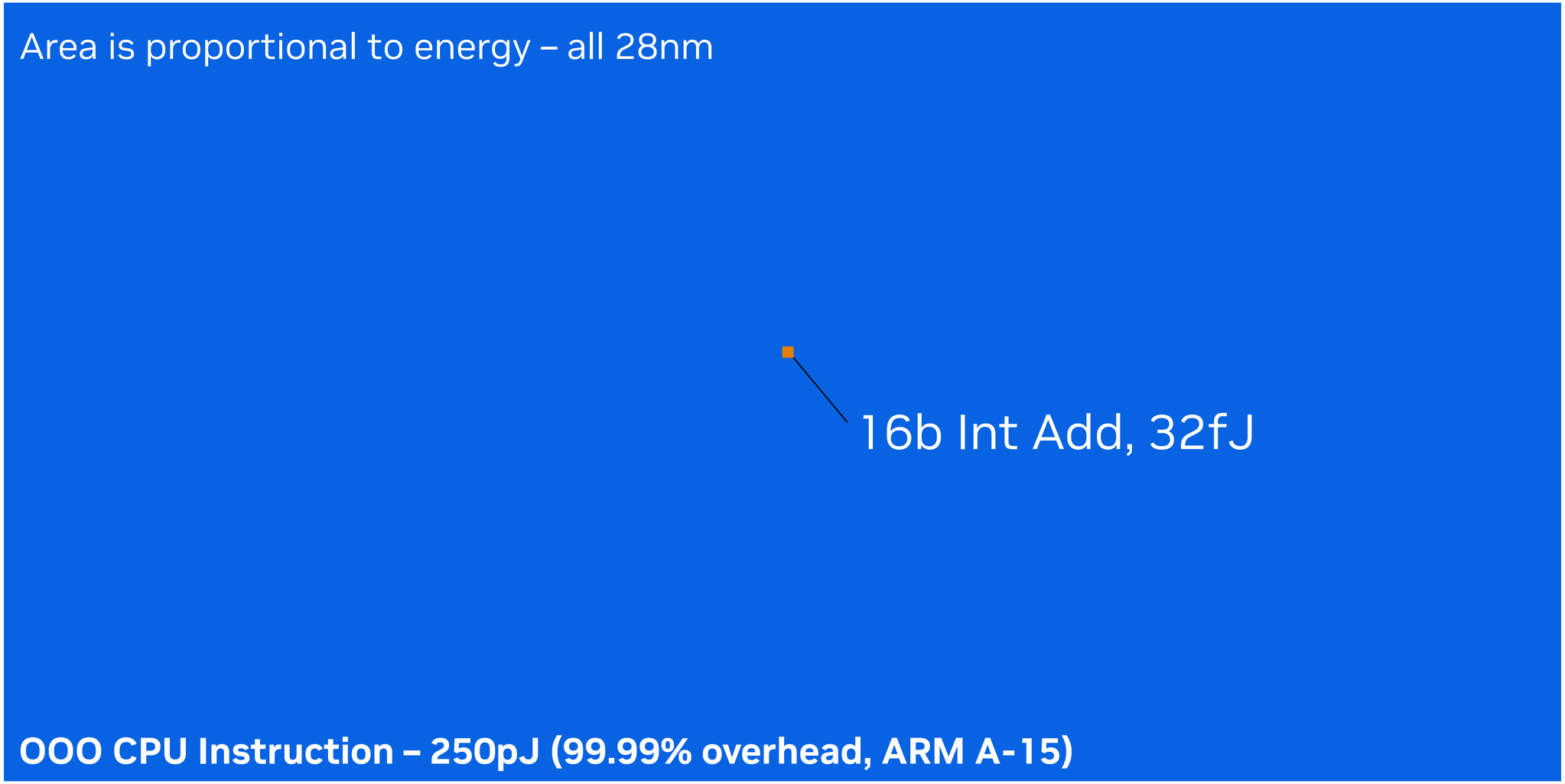
- Communication (~100fJ/b-mm on-chip)
- Memory (~50fJ/b for small RAM)
- Operations (~1fJ/b for add)

A Prescription

- Do
  - Less (fewer operations)
  - It With smaller data (movement cheaper, math cheaper[2])
  - Locally (less movement)
  - Combinationally (flops burn energy)
  - It sparsely

Do Less

Do Less Overhead



Evangelos Vasilakis. 2015. *An Instruction Level Energy Characterization of Arm Processors.* Foundation of Research and Technology Hellas, Inst. of Computer Science, Tech. Rep. FORTH-ICS/TR-450 (2015)

Specialized Instructions Amortize Overhead

Operation	Energy**	Overhead*
HFMA	1.5pJ	2000%
HDP4A	6.0pJ	500%
HMMA	110pJ	22%
IMMA	160pJ	16%

*Overhead is instruction fetch, decode, and operand fetch – 30pJ
**Energy numbers from 45nm process

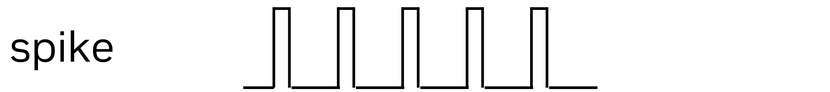
Communication-Efficient Algorithms

Don’t minimize big-O ops, minimize cost.

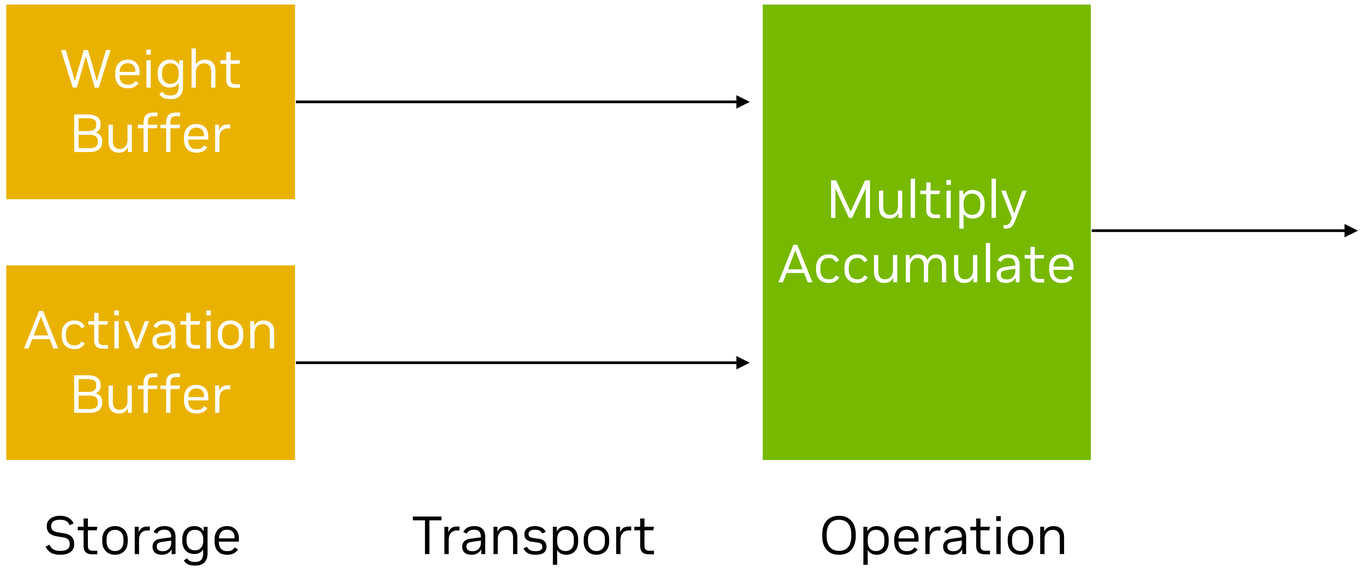
An add is worth 10um of movement

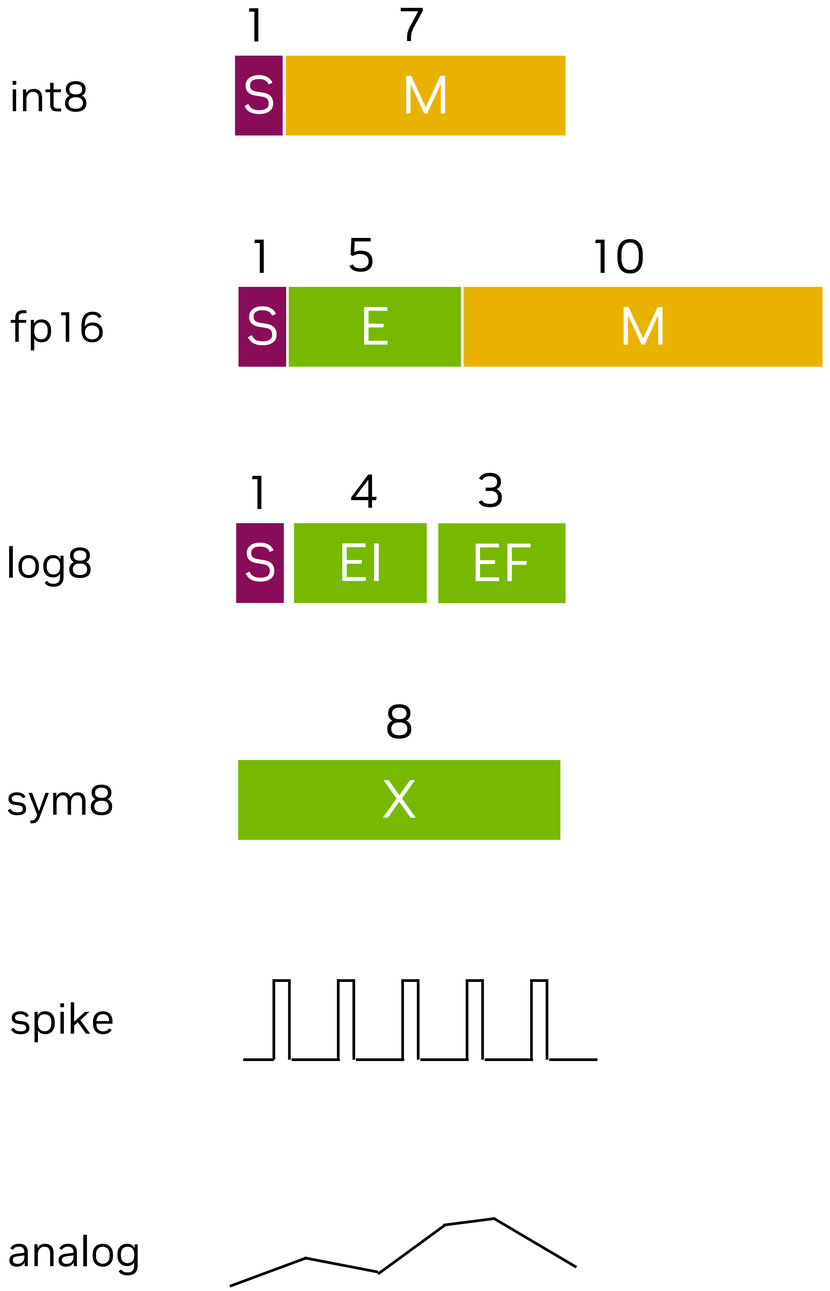
Dally, William. "On the model of computation: point." *Communications of the ACM* 65.9 (2022): 30-32.

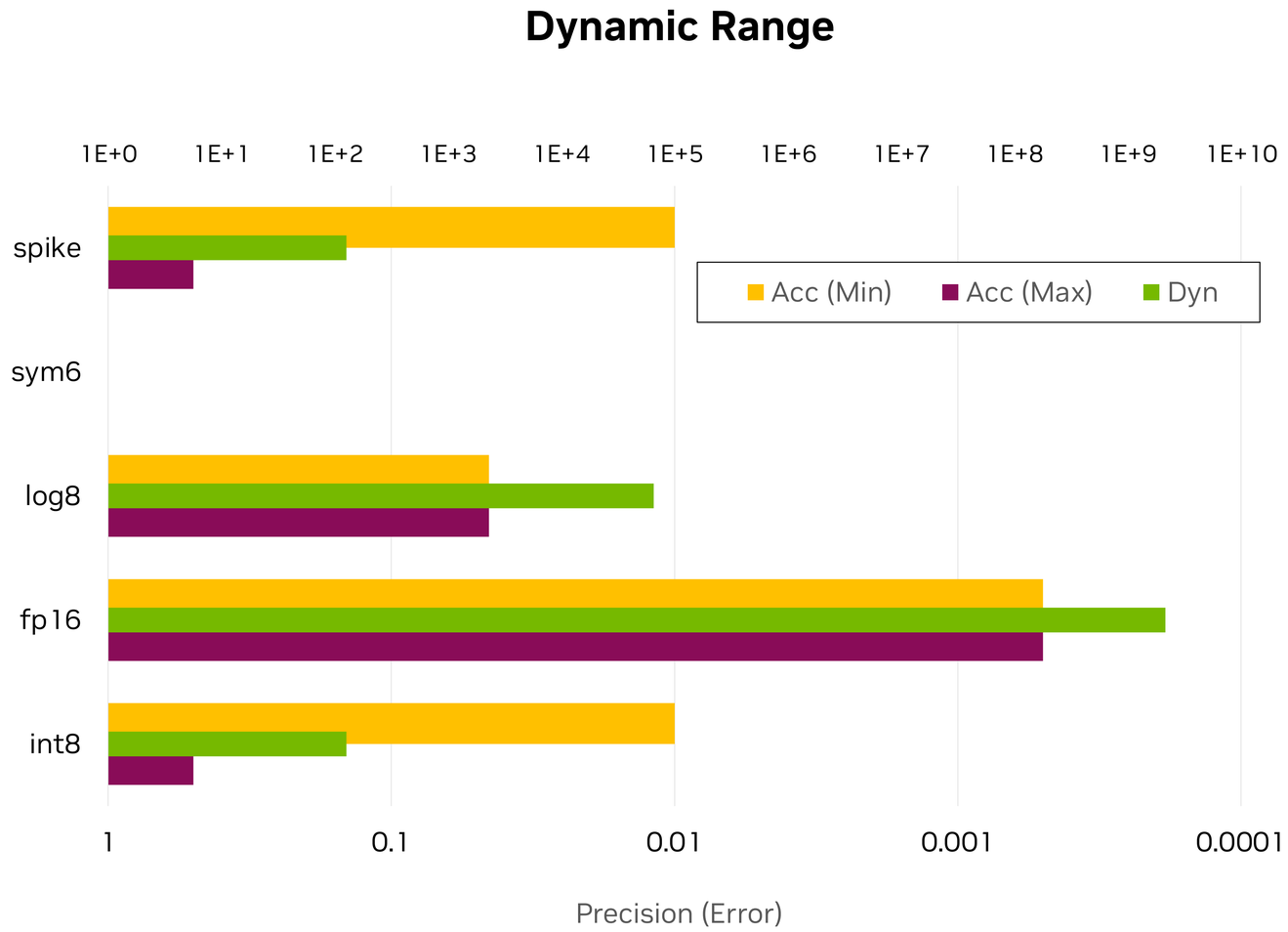
Do it with Smaller Data



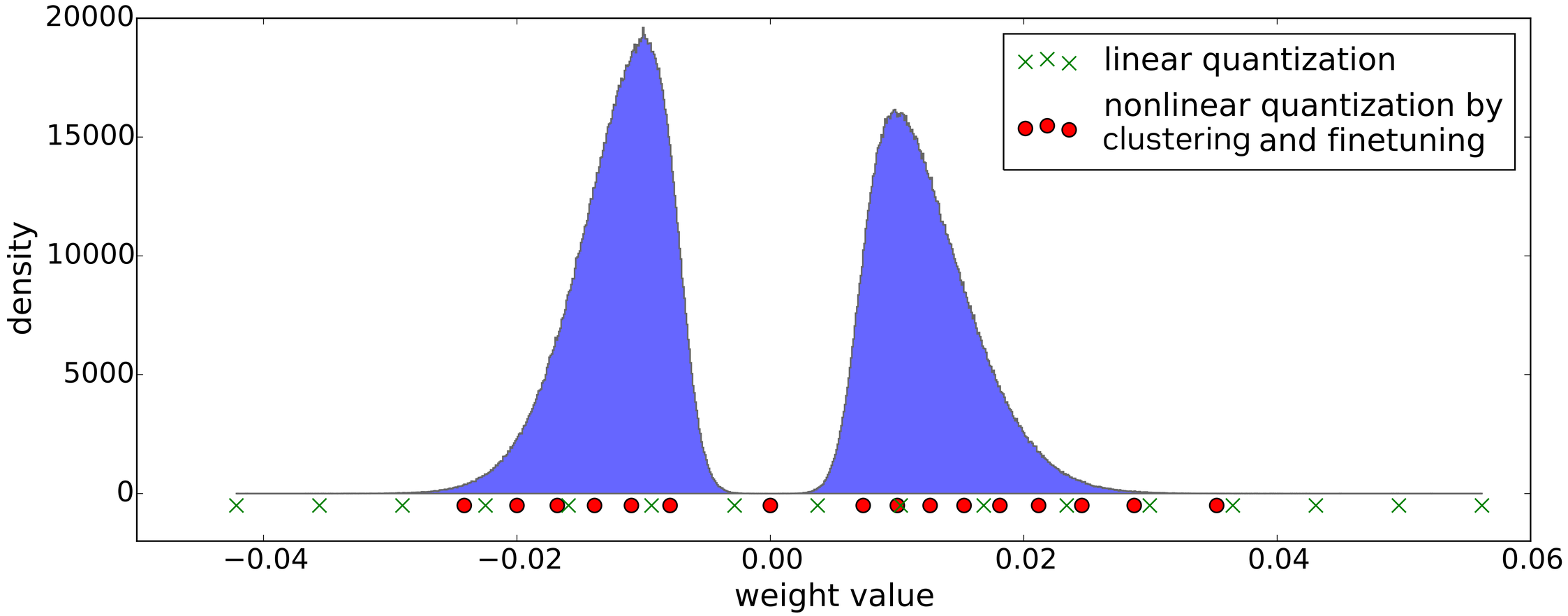
- Attributes:
  - Cost
    - Operation energy
    - Movement energy
  - Accuracy
    - Dynamic range
    - Precision (error)



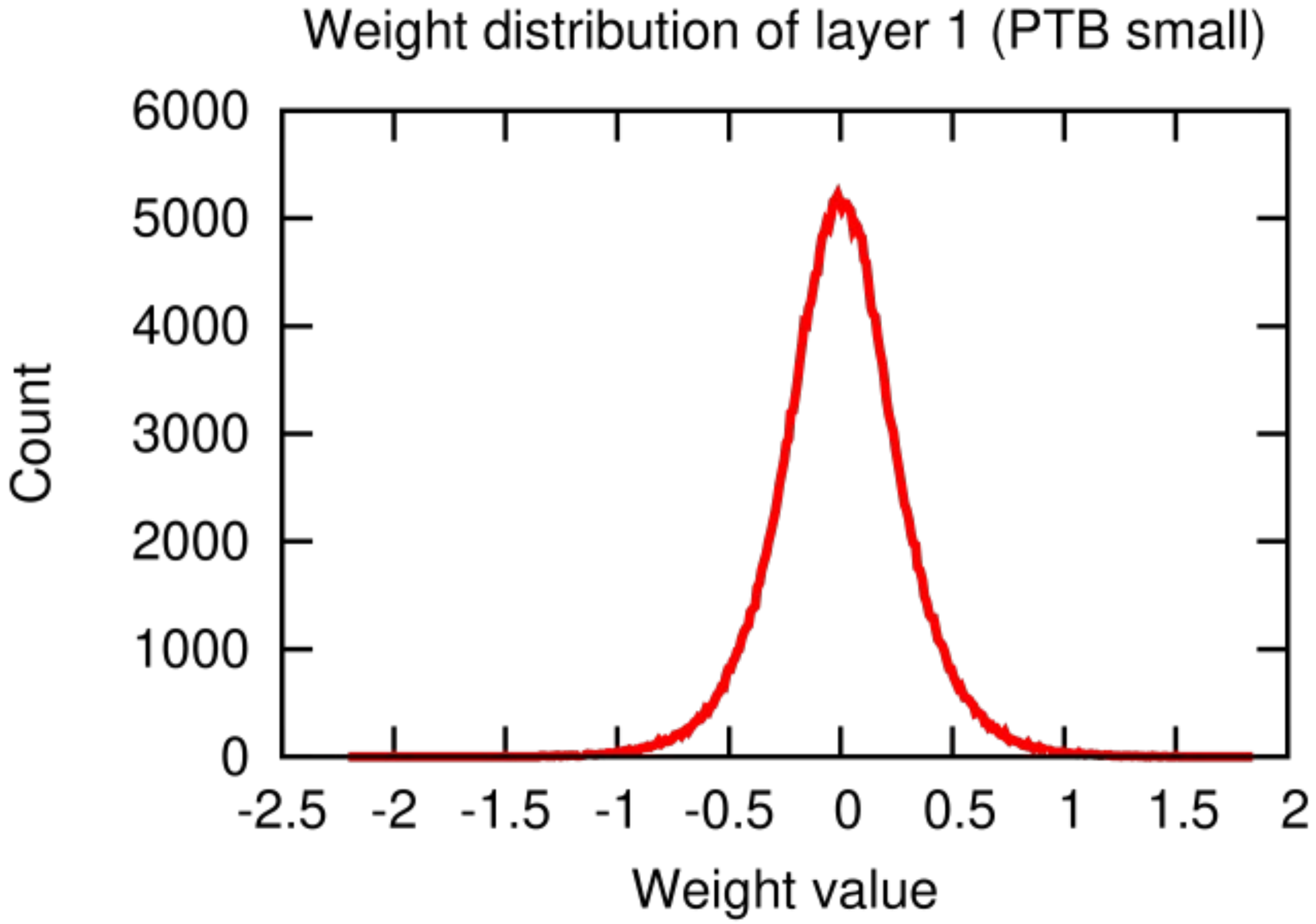

int8
1
7
S
M
fp16
1
5
10
S
E
M
log8
1
4
3
S
EI
EF
sym8
8
X
spike
analog


Dynamic Range
1E+0
1E+1
1E+2
1E+3
1E+4
1E+5
1E+6
1E+7
1E+8
1E+9
1E+10
spike
sym6
log8
fp16
int8
Acc (Min)
Acc (Max)
Dyn
1
0.1
0.01
0.001
0.0001
Precision (Error)

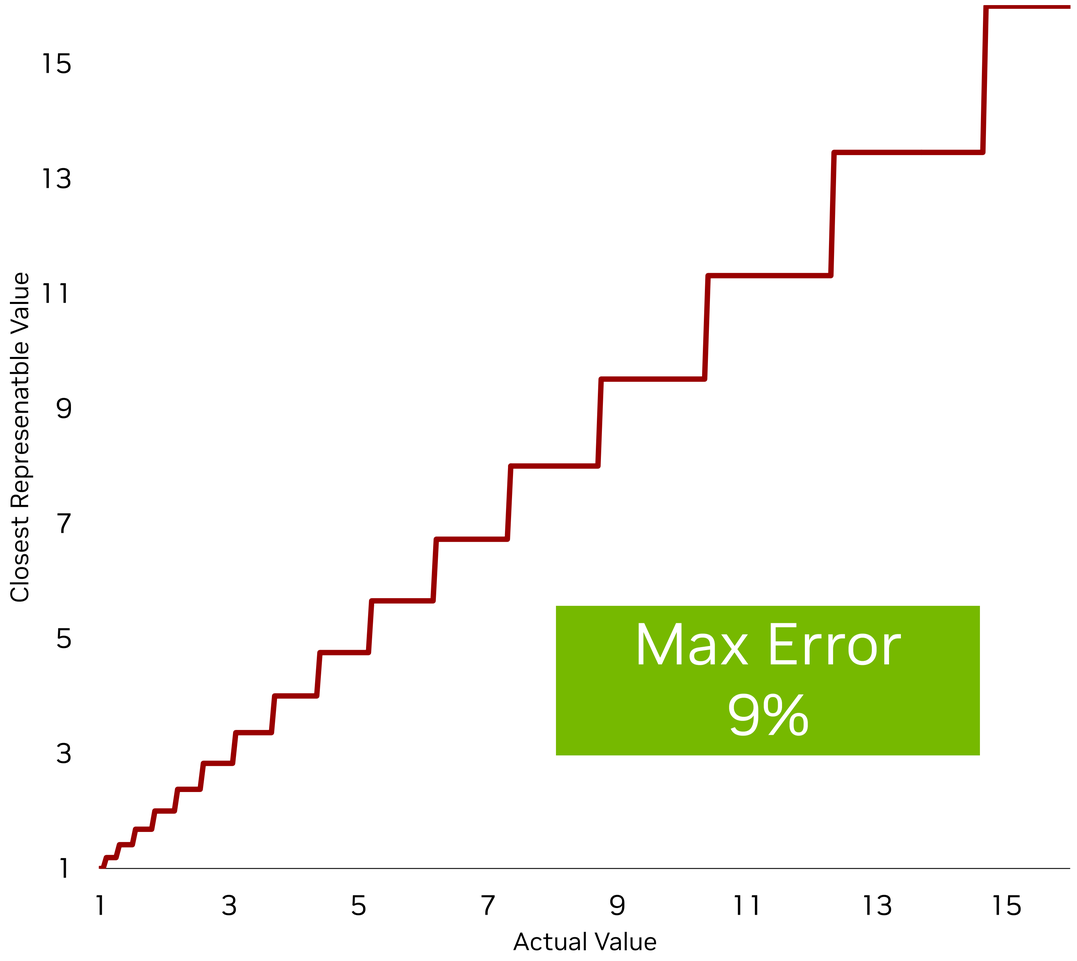
Symbol Representation (Codebook)



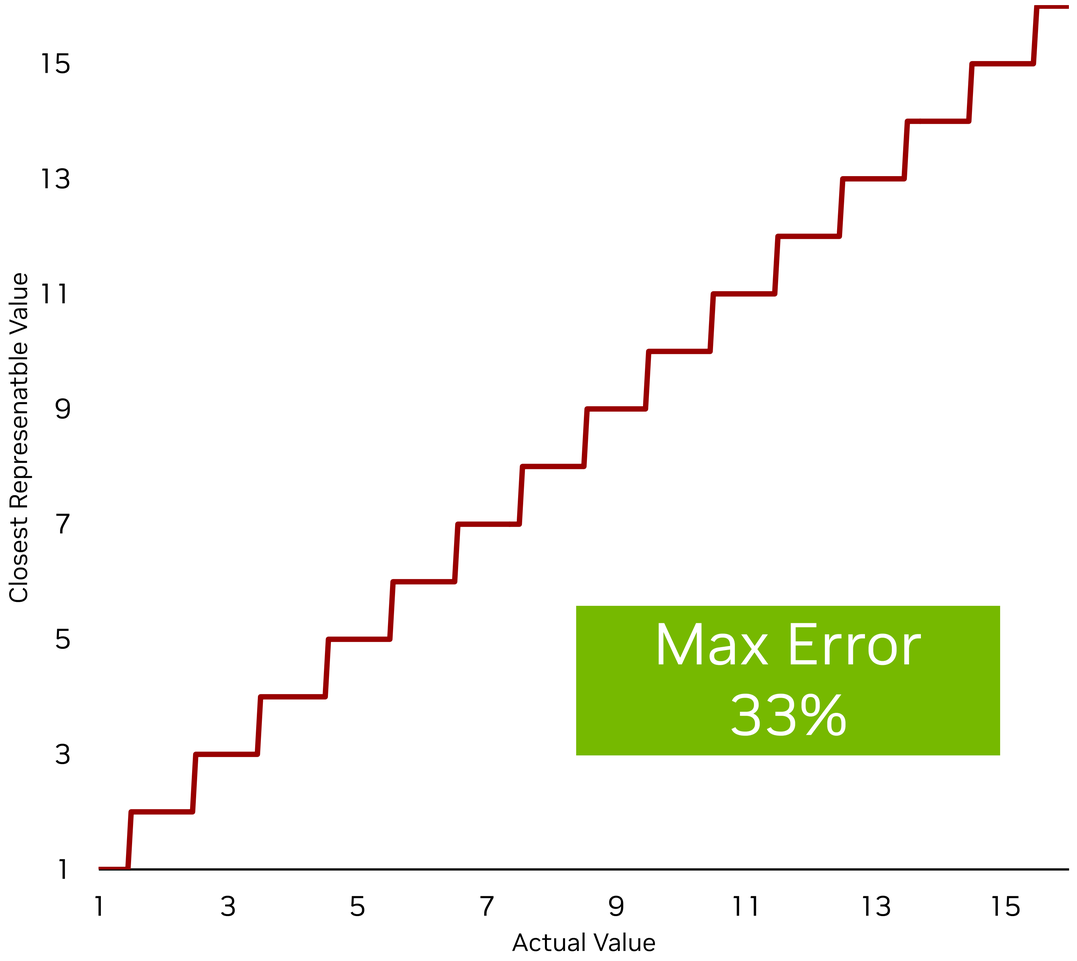
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv 2015


Weight distribution of layer 1 (PTB small)
Count
6000
5000
4000
3000
2000
1000
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Weight value

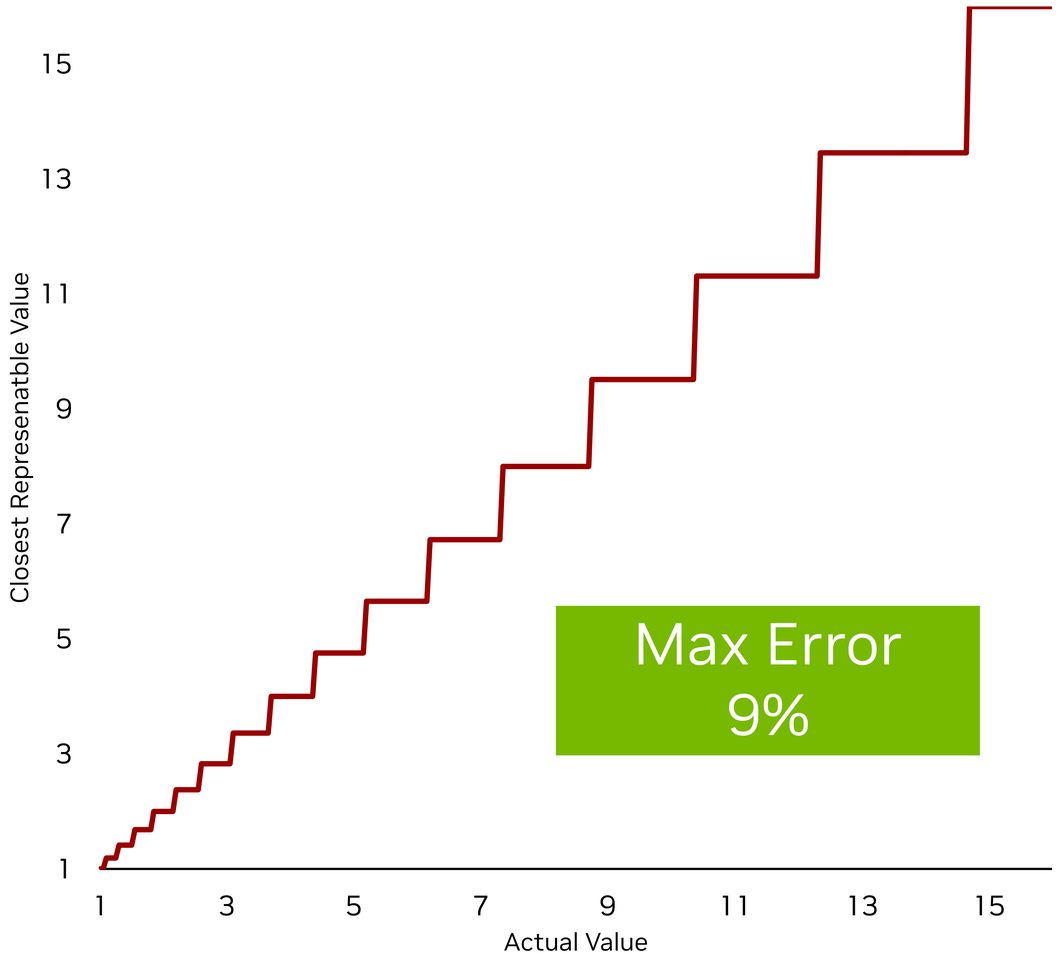
4-bit Log Representation (L2.2)



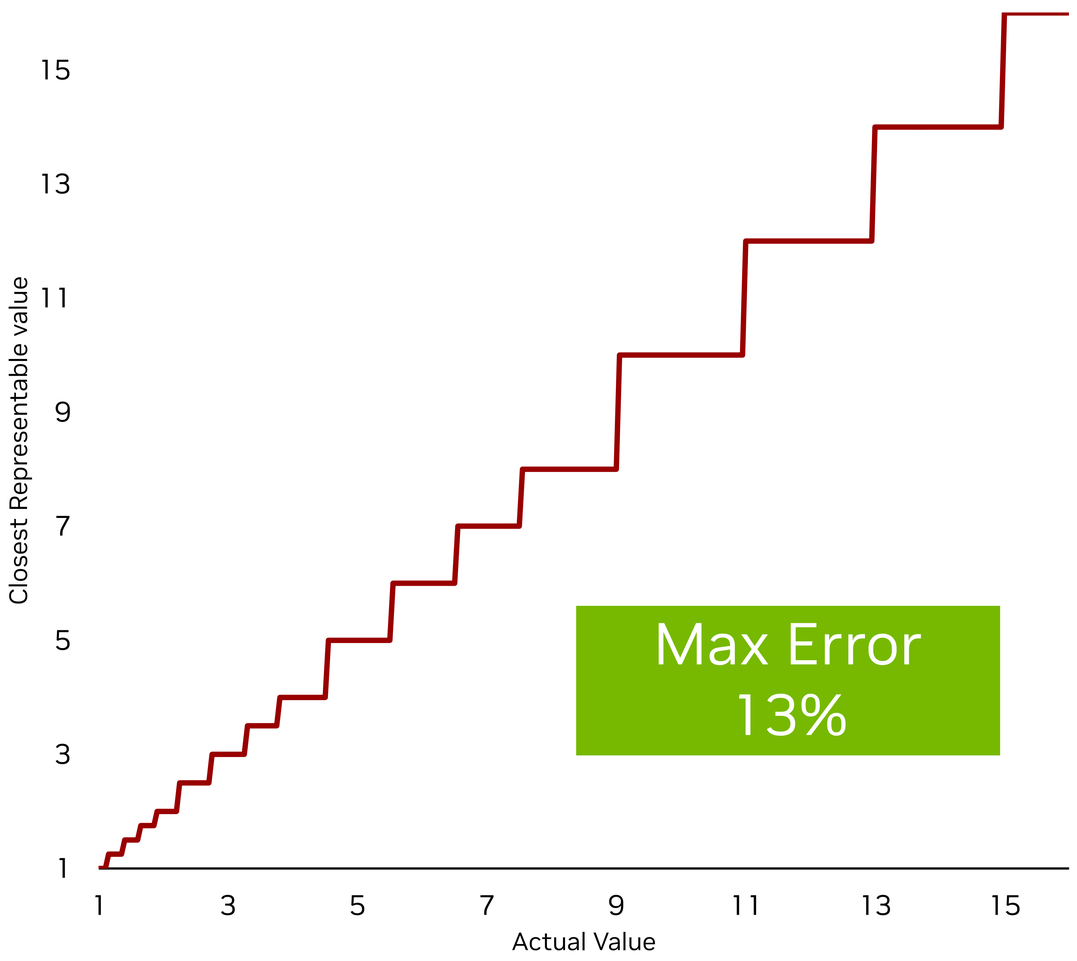
4-bit Integer Representation (Int4)



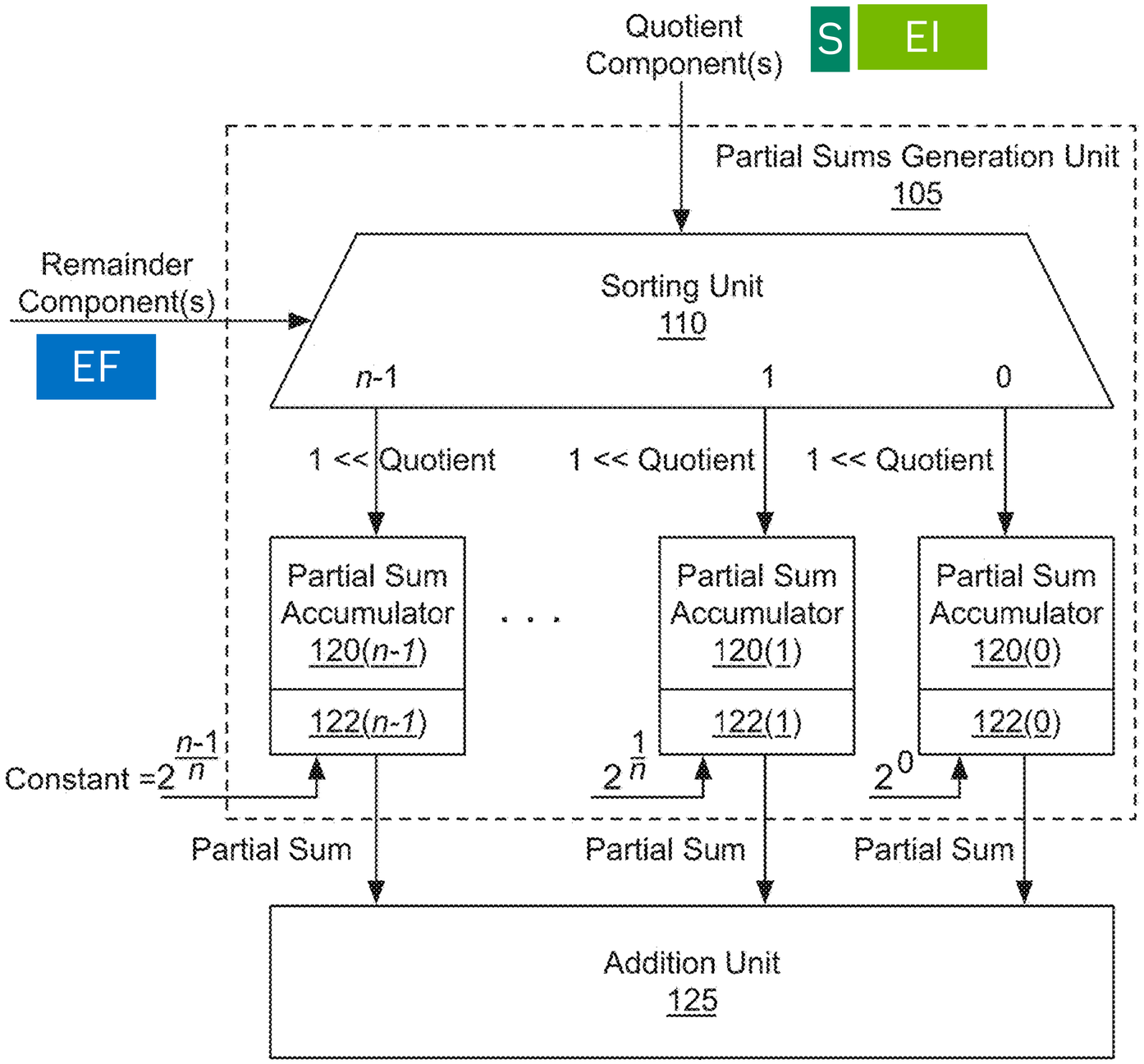
4-bit Log Representation (L2.2)



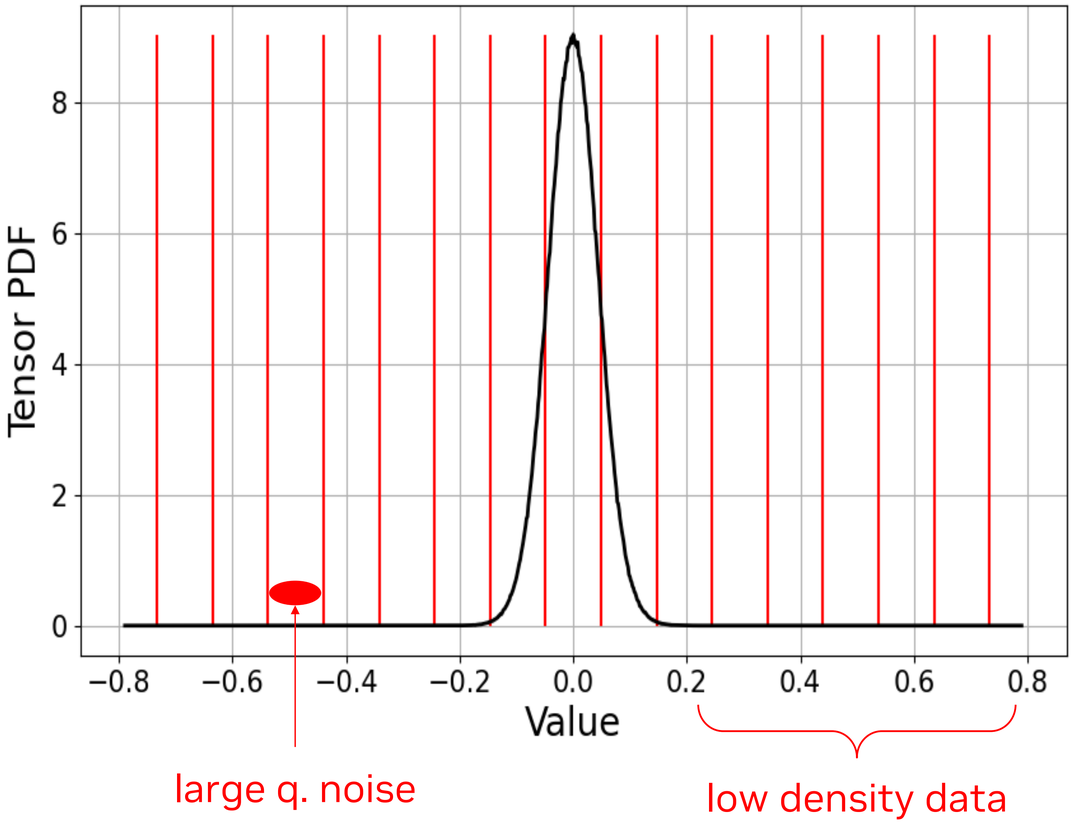
FP2.2

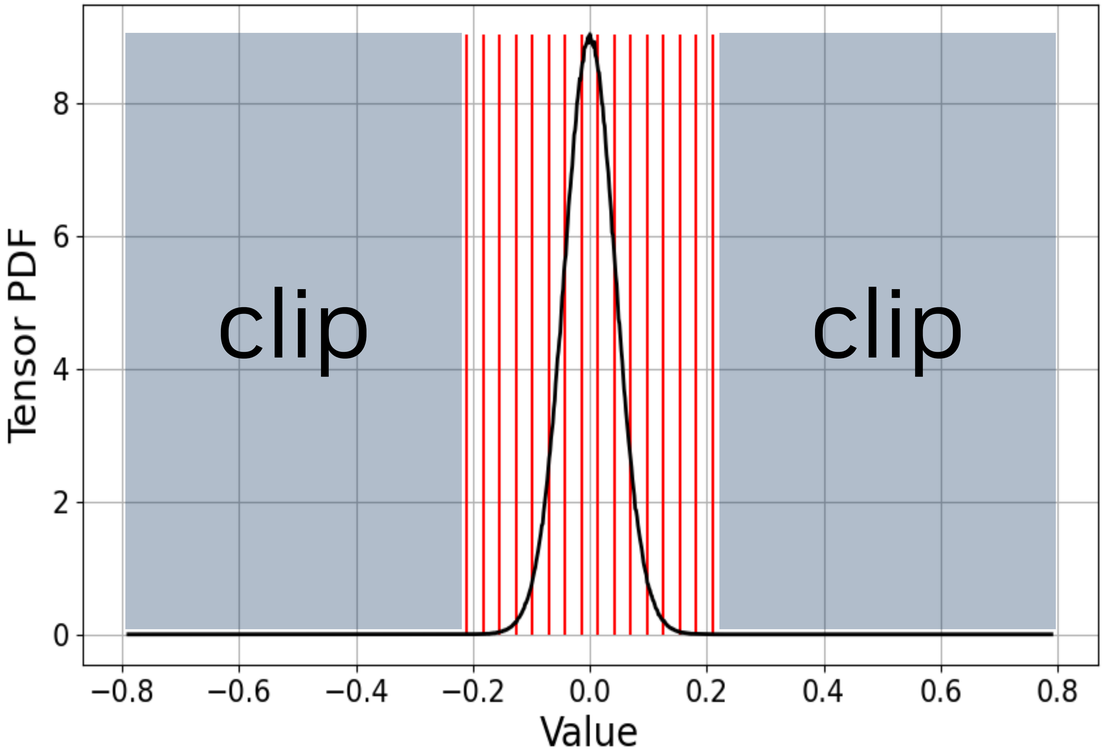


- Log Numbers
- Multiplies are cheap – just an add
- Adds are hard – convert to integer, add, convert back
  - Fractional part of log is a lookup
  - Integer part of log is a shift
- Can factor the lookup outside the summation
  - Only convert back after summation (and NLF)


Quotient Component(s)
S
EI
Partial Sums Generation Unit
105
Remainder Component(s)
EF
Sorting Unit
110
n-1
1
0
1 << Quotient
1 << Quotient
1 << Quotient
Partial Sum Accumulator 120(n-1)
Partial Sum Accumulator 120(1)
Partial Sum Accumulator 120(0)
122(n-1)
122(1)
122(0)
Constant = 2^(n-1)/n
2^1/n
2^0
Partial Sum
Partial Sum
Partial Sum
Addition Unit
125


1
4
3
S
EI
EF

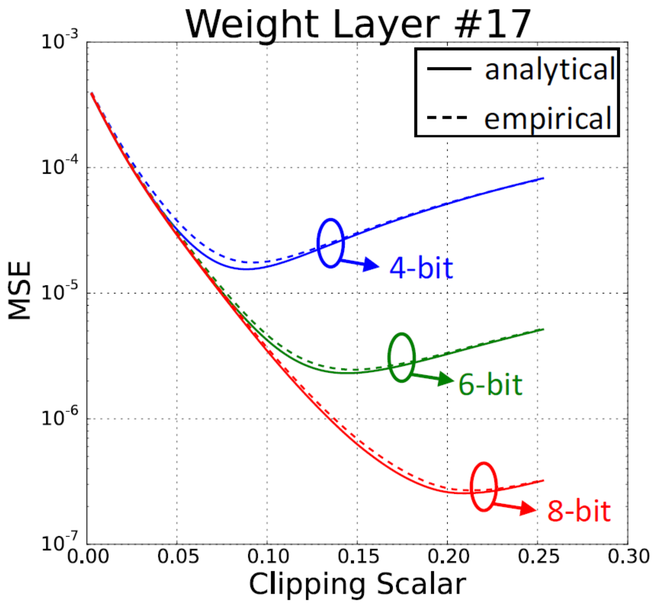


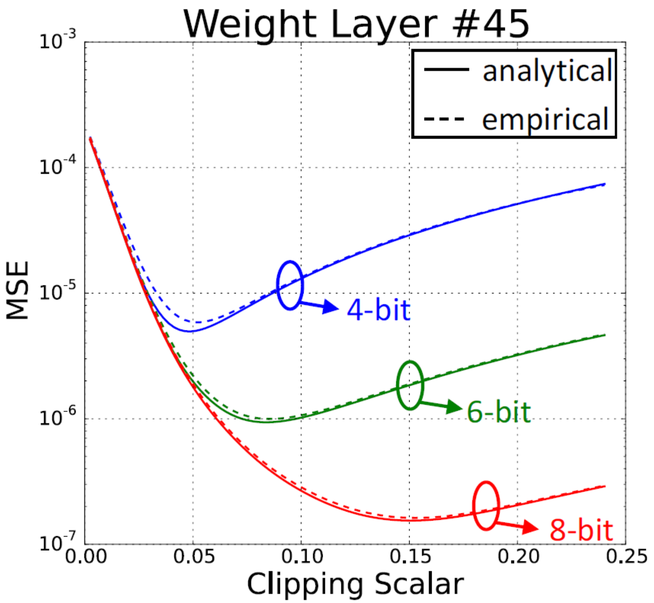


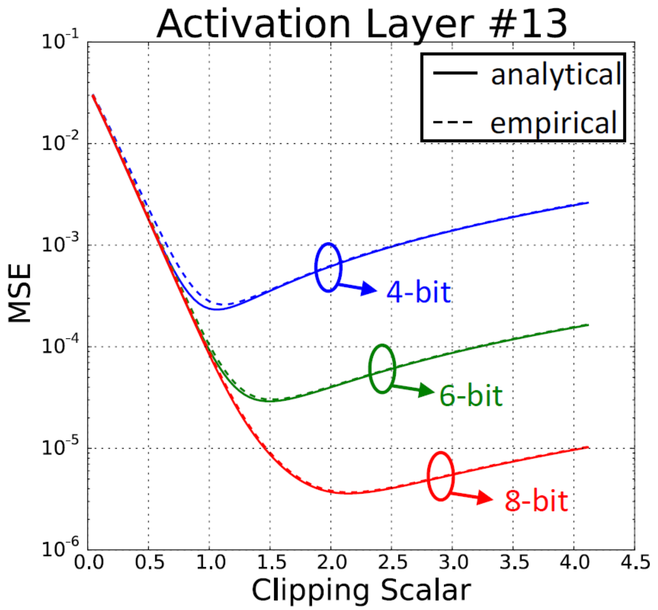
Sakr, Charbel, et al. "Optimal clipping and magnitude-aware differentiation for improved quantization-aware training." *International Conference on Machine Learning*. PMLR, 2022

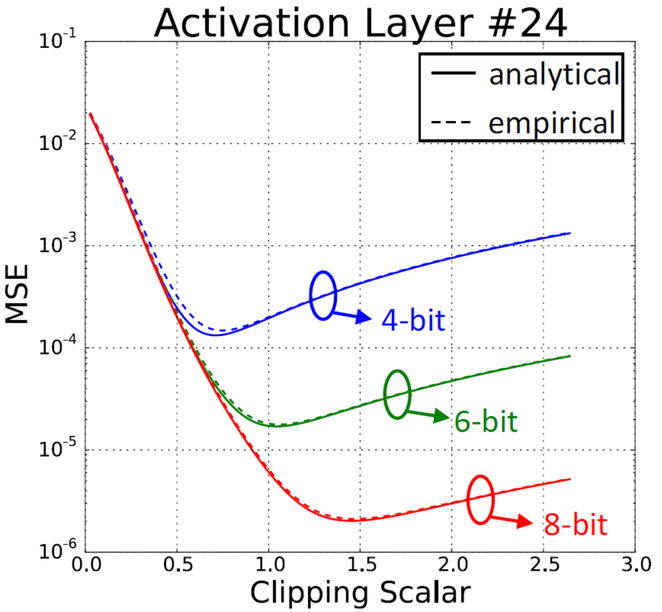
$$J = \frac{4^{-B}}{3} s^2 \int_0^s f_{|X|}(x)dx + \int_s^\infty (s-x)^2 f_{|X|}(x)dx$$

$$s_{n+1} = \frac{\boldsymbol{E}\left[|X| \cdot \mathbf{1}_{\{|X|>s_n\}}\right]}{\frac{4^{-B}}{3}\boldsymbol{E}\left[\mathbf{1}_{\{|X|<s_n\}}\right] + \boldsymbol{E}\left[\mathbf{1}_{\{|X|>s_n\}}\right]}$$

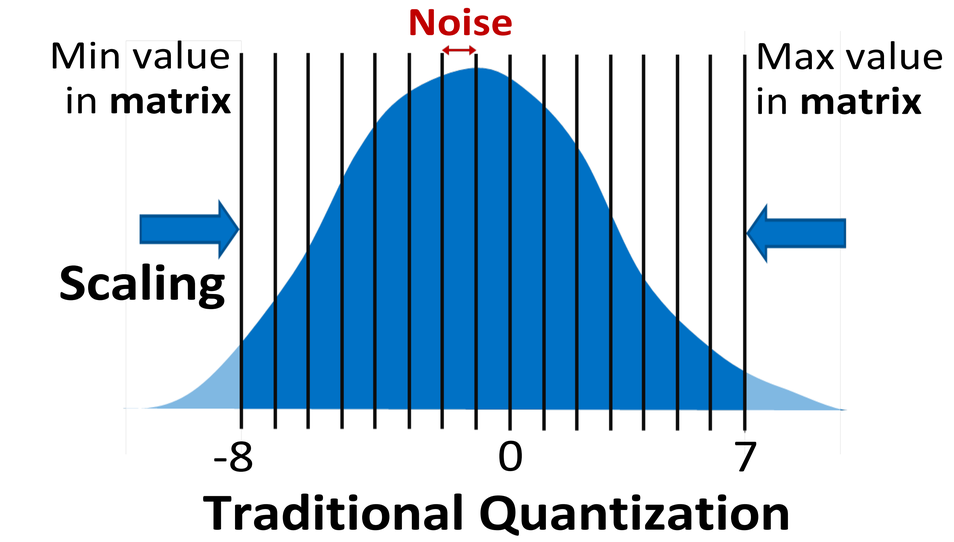


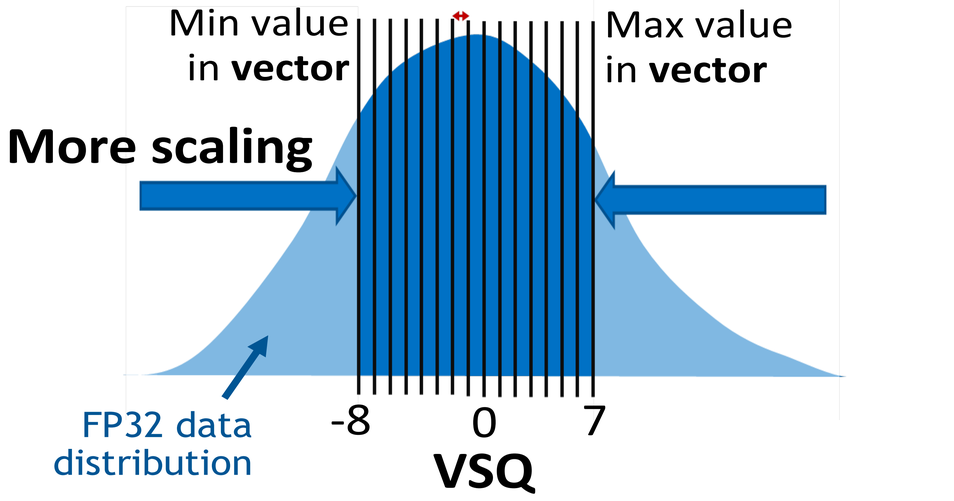




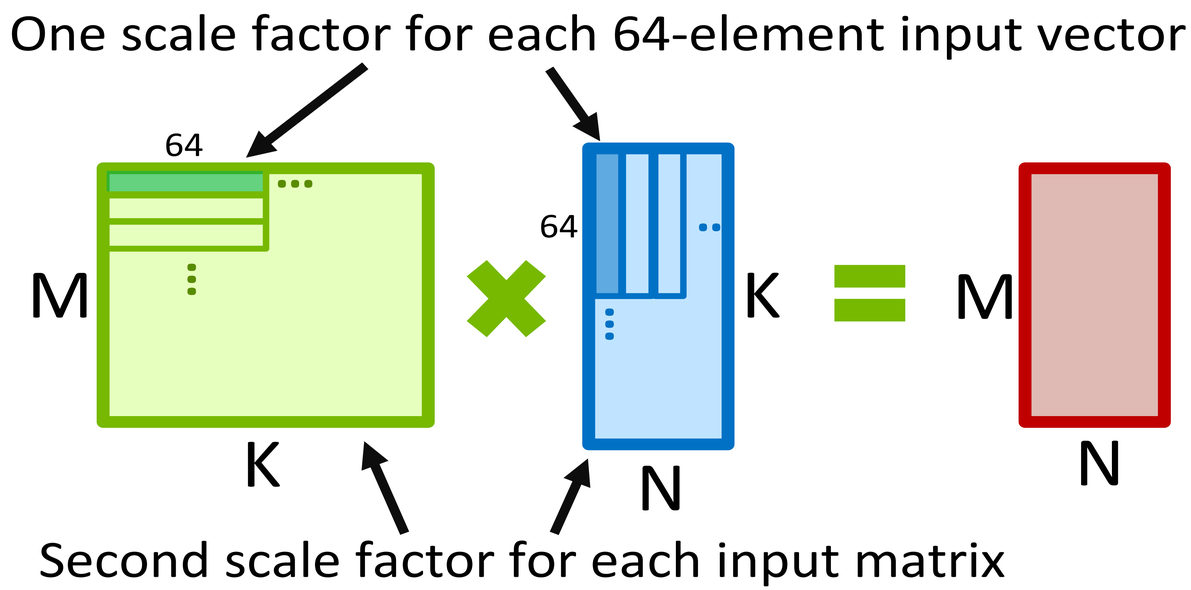


INT4 Quantization





VSQ Scale Factors



Traditional Quantization	VSQ
One scale factor per matrix	Two scale factors: one per vector, one per matrix
High quantization noise	Reduced quantization noise

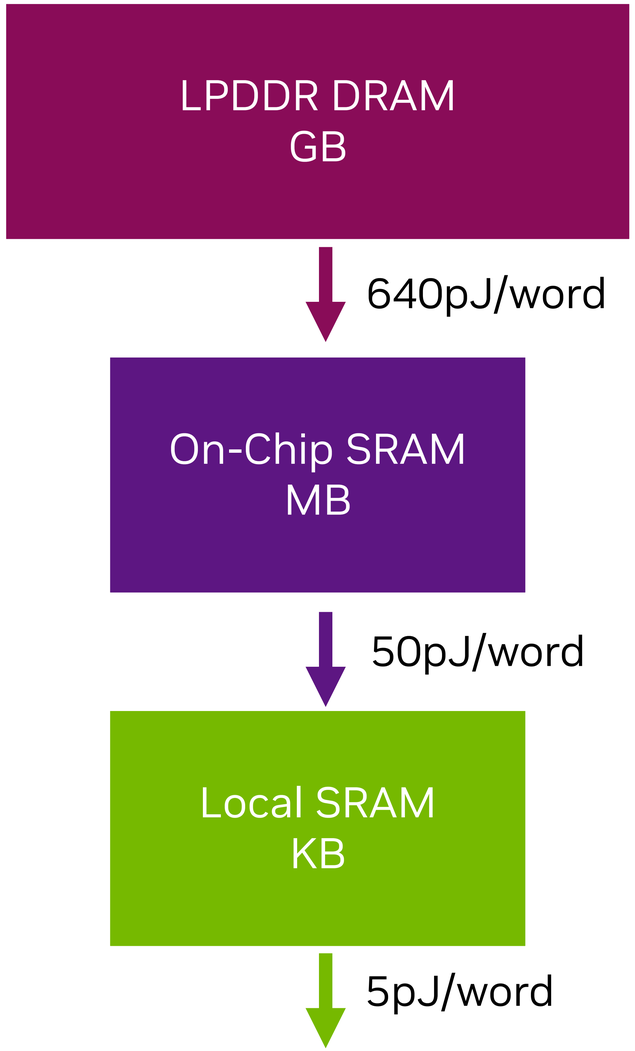
Dai, Steve, et al. "Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference." *Proceedings of Machine Learning and Systems* 3 (2021): 873-884.

Do it Locally

Do it Locally

Cost of an add (1fJ/bit) = Cost of going 10um.

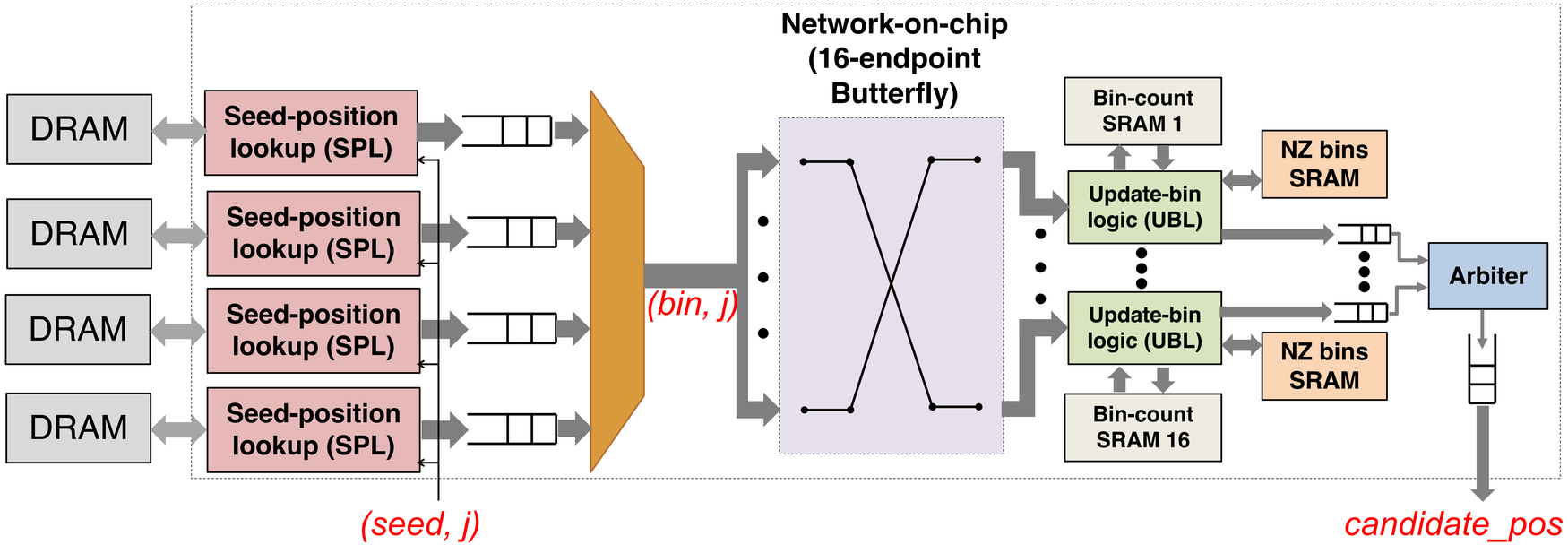
The Importance of Staying Local

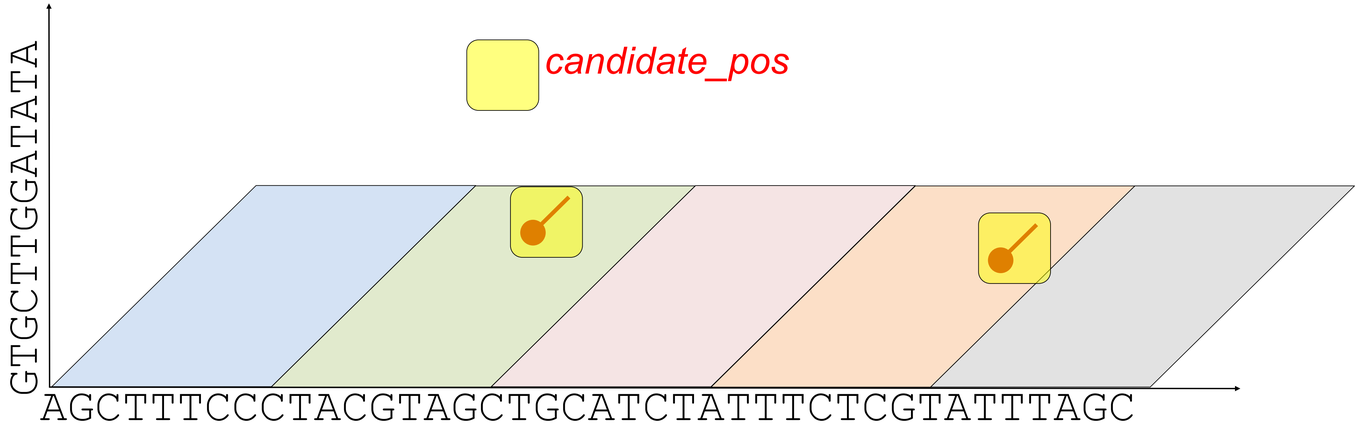


Message-Driven Processing
One Communication, Many Operations

One traversal of network

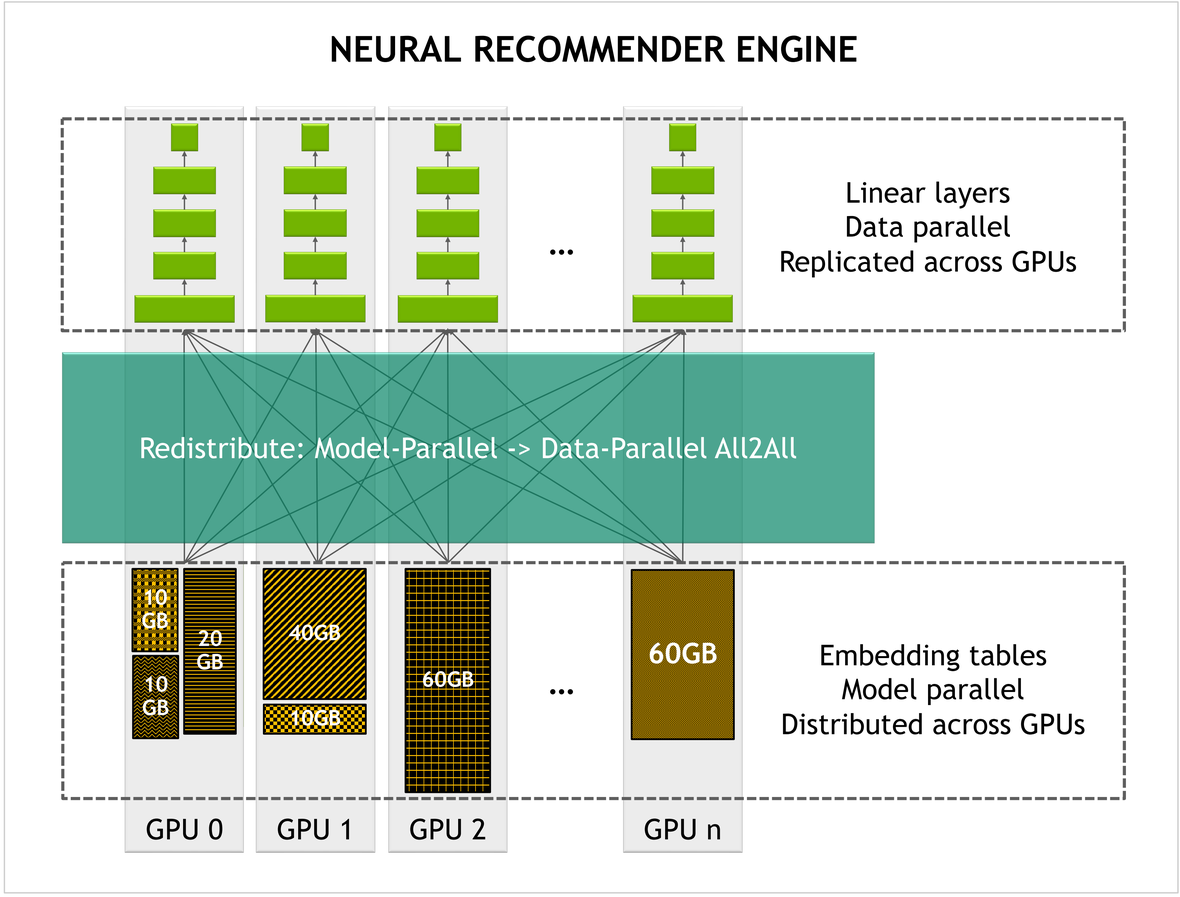
- Access hash table
- Increment bin (RMW)
- If it was zero, append to NZ bins
- If over threshold, append to output queue

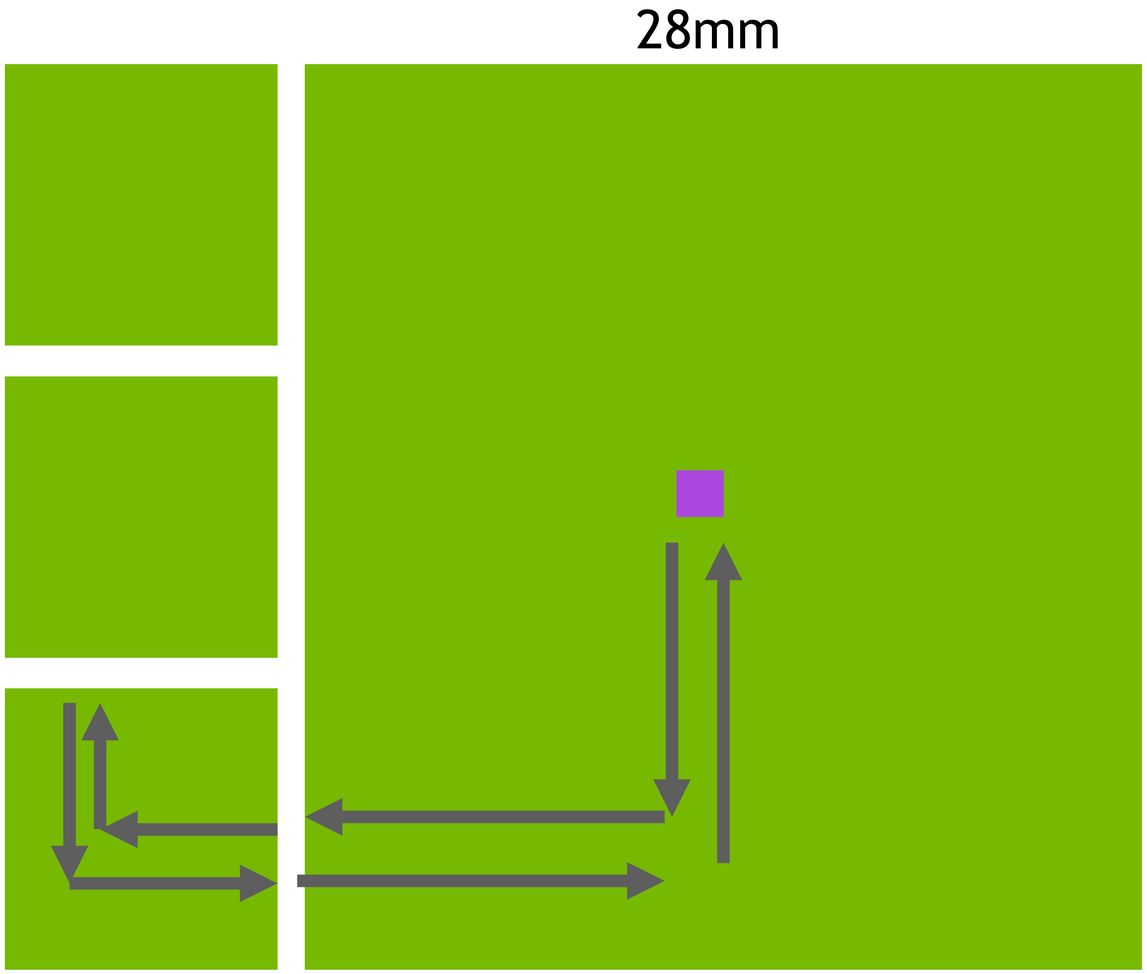




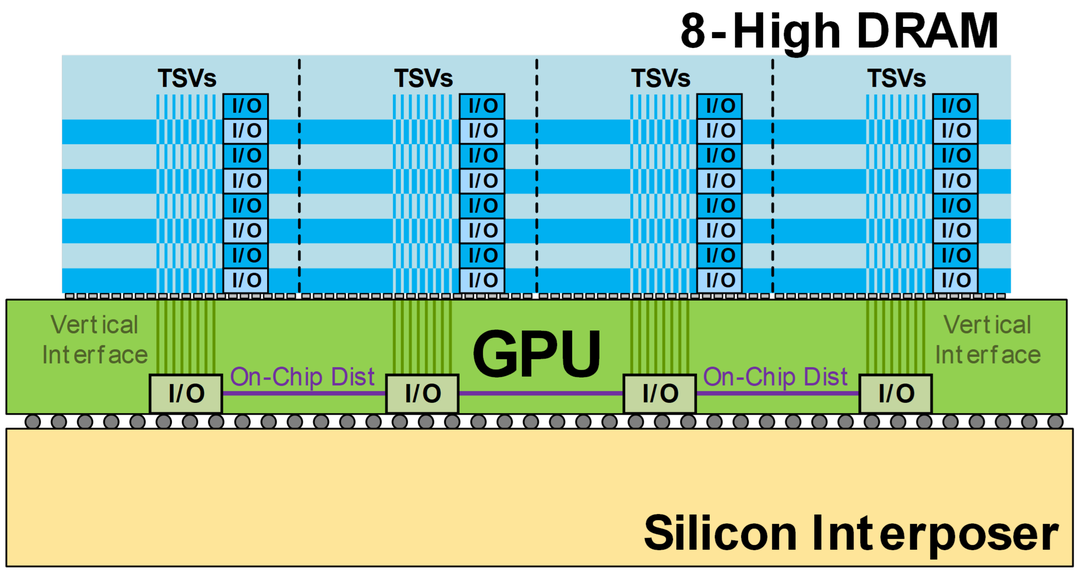
Turakhia, Yatish, Gill Bejerano, and William J. Dally. "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly." *ACM SIGPLAN Notices* 53.2 (2018): 199-213.

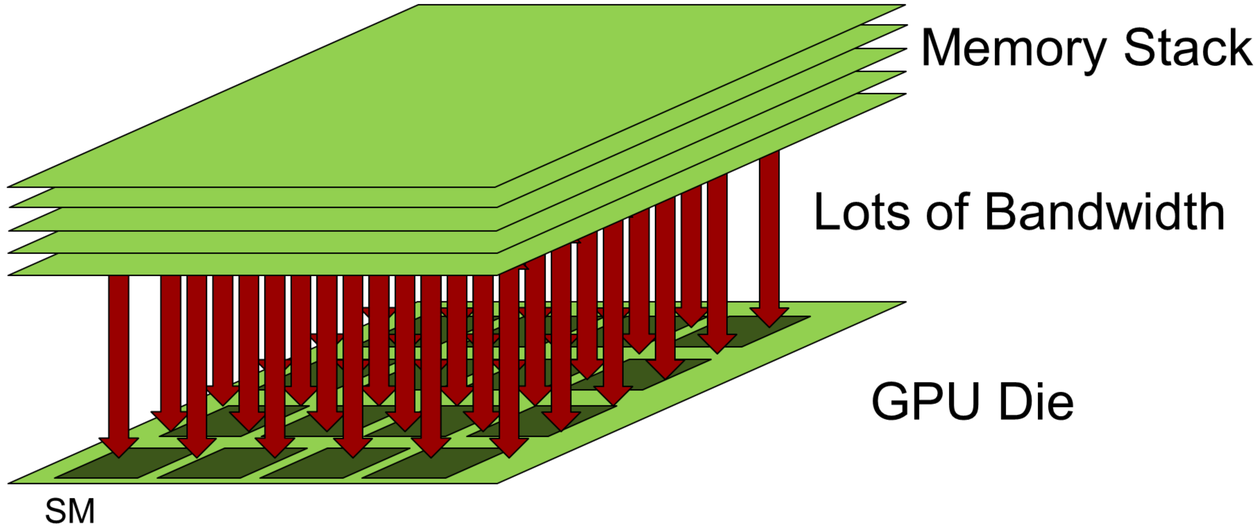
Many-Hot Recommenders





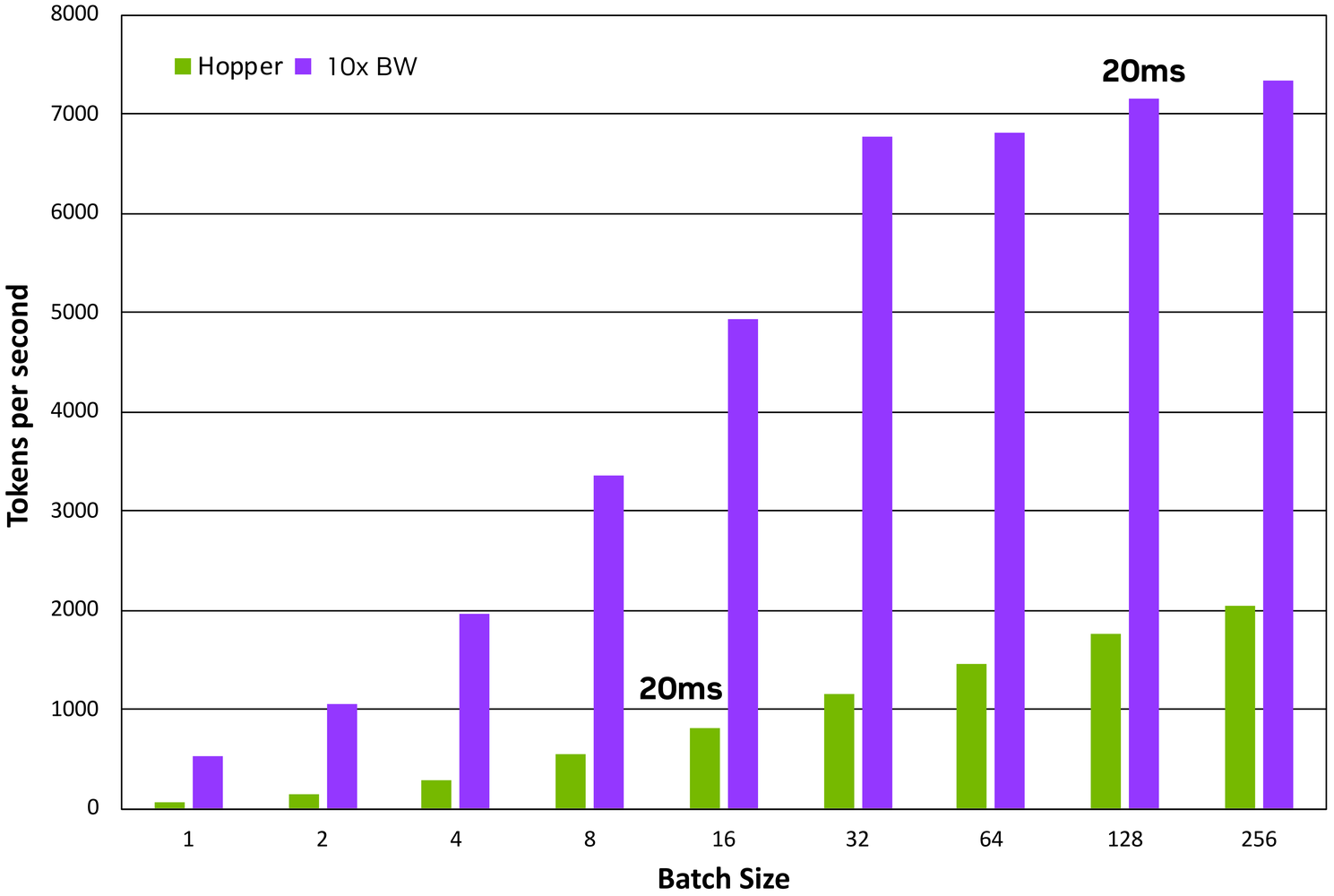
- 48mm round trip on GPU die
  - 4.8pJ/b @ 100fJ/b-mm
- 16mm round trip on DRAM die
  - 1.6pJ/b
  - Part of 5pJ/b access


8-High DRAM
TSVs
TSVs
TSVs
TSVs
I/O
Vertical Interface
On-Chip Dist
GPU
On-Chip Dist
Vertical Interface
Silicon Interposer


Memory Stack
Lots of Bandwidth
GPU Die
SM

Large Language Model (LLM) Inference

Megatron 20B parameter model

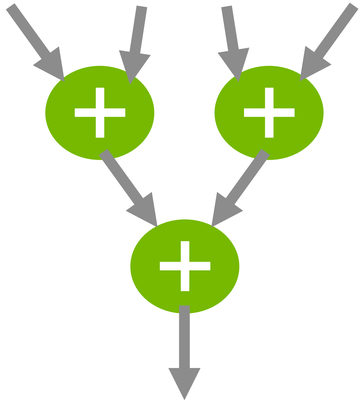


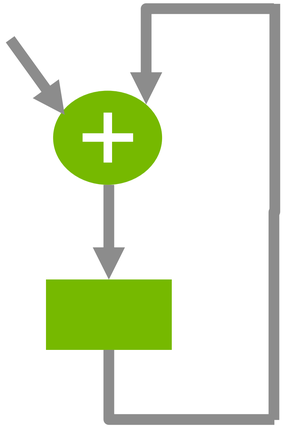
- 3.6x faster at large batch sizes
- 8.9x faster at 20ms/token

Do it Combinationally

Do it Combinationally

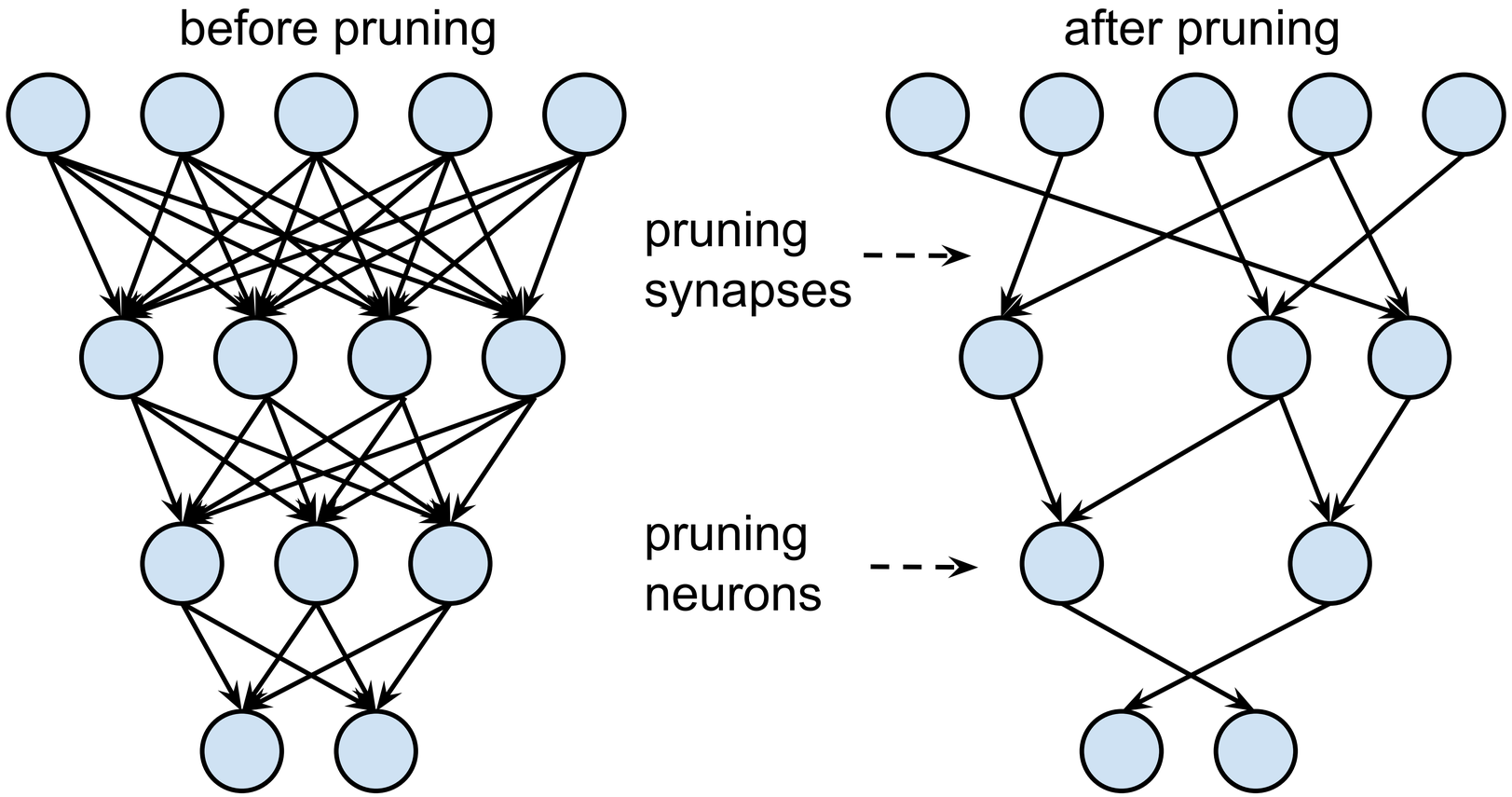
Cost of an add ~ Cost of a flip flop (1fJ/b)





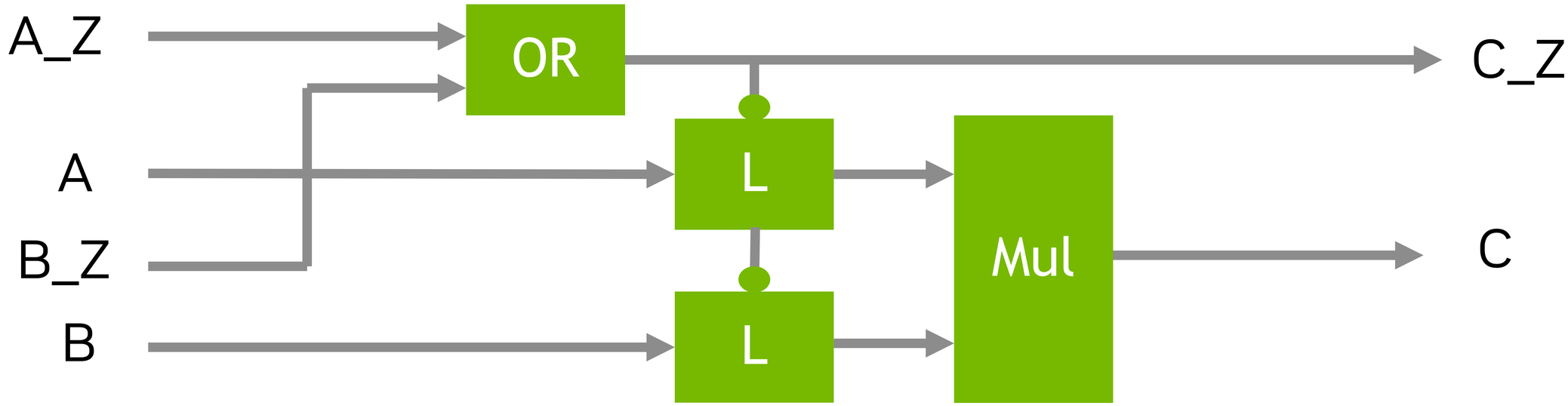
Do it Sparsely

Pruning



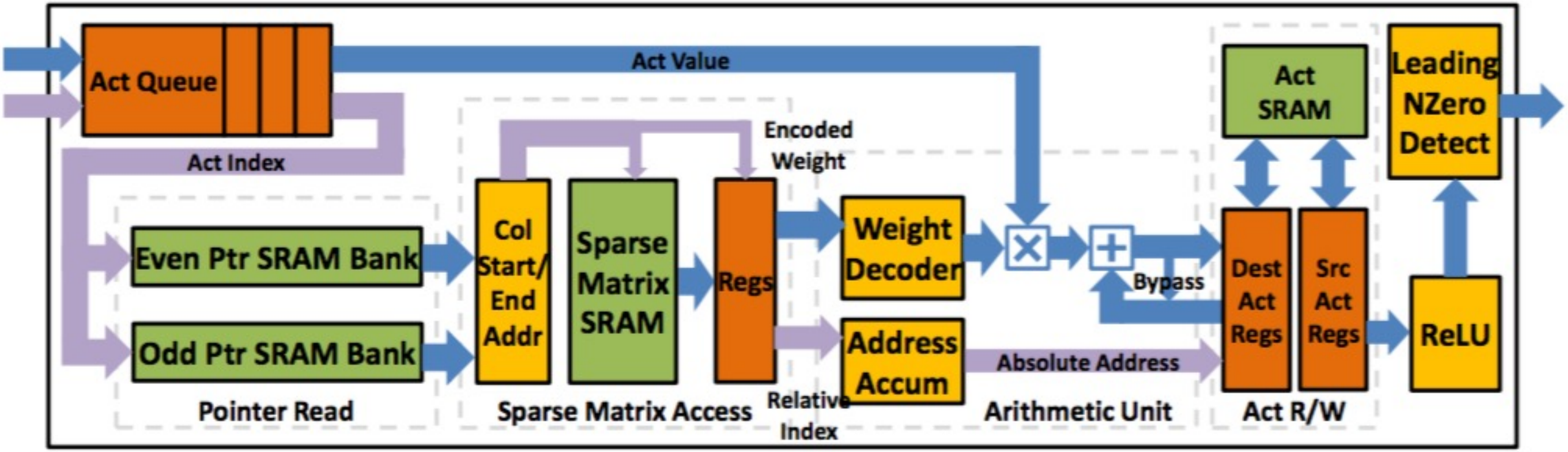
Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

Zero Gating

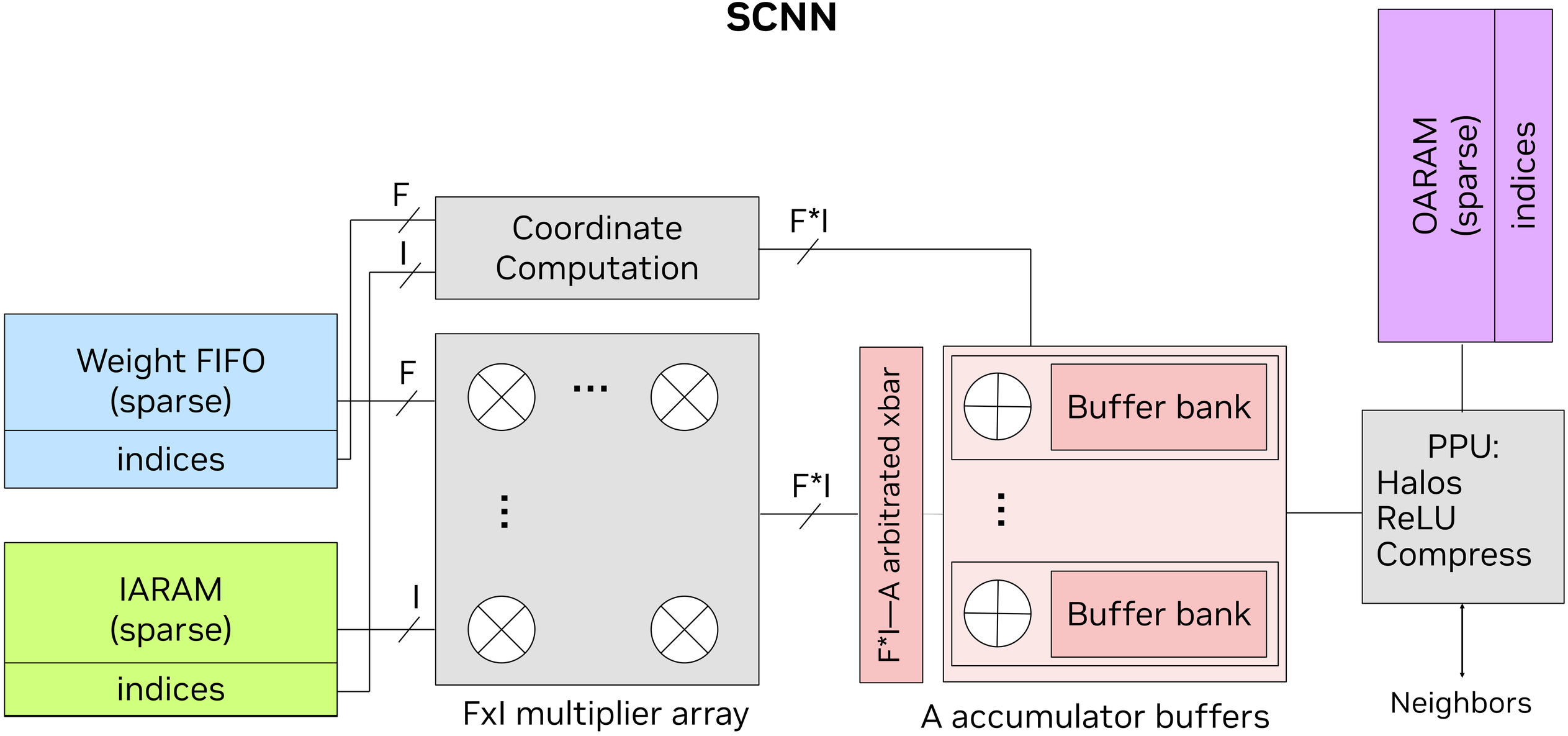


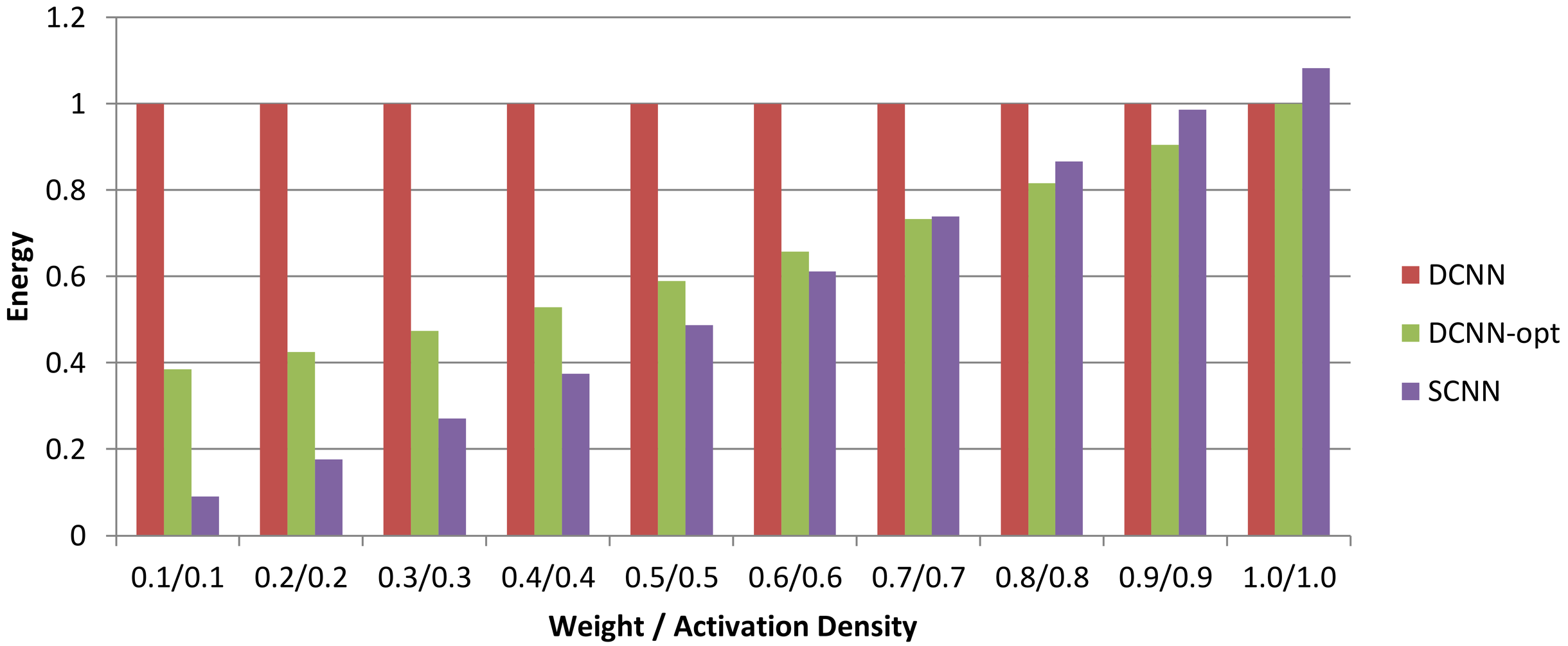
Implemented in NVDLA 2014

EIE

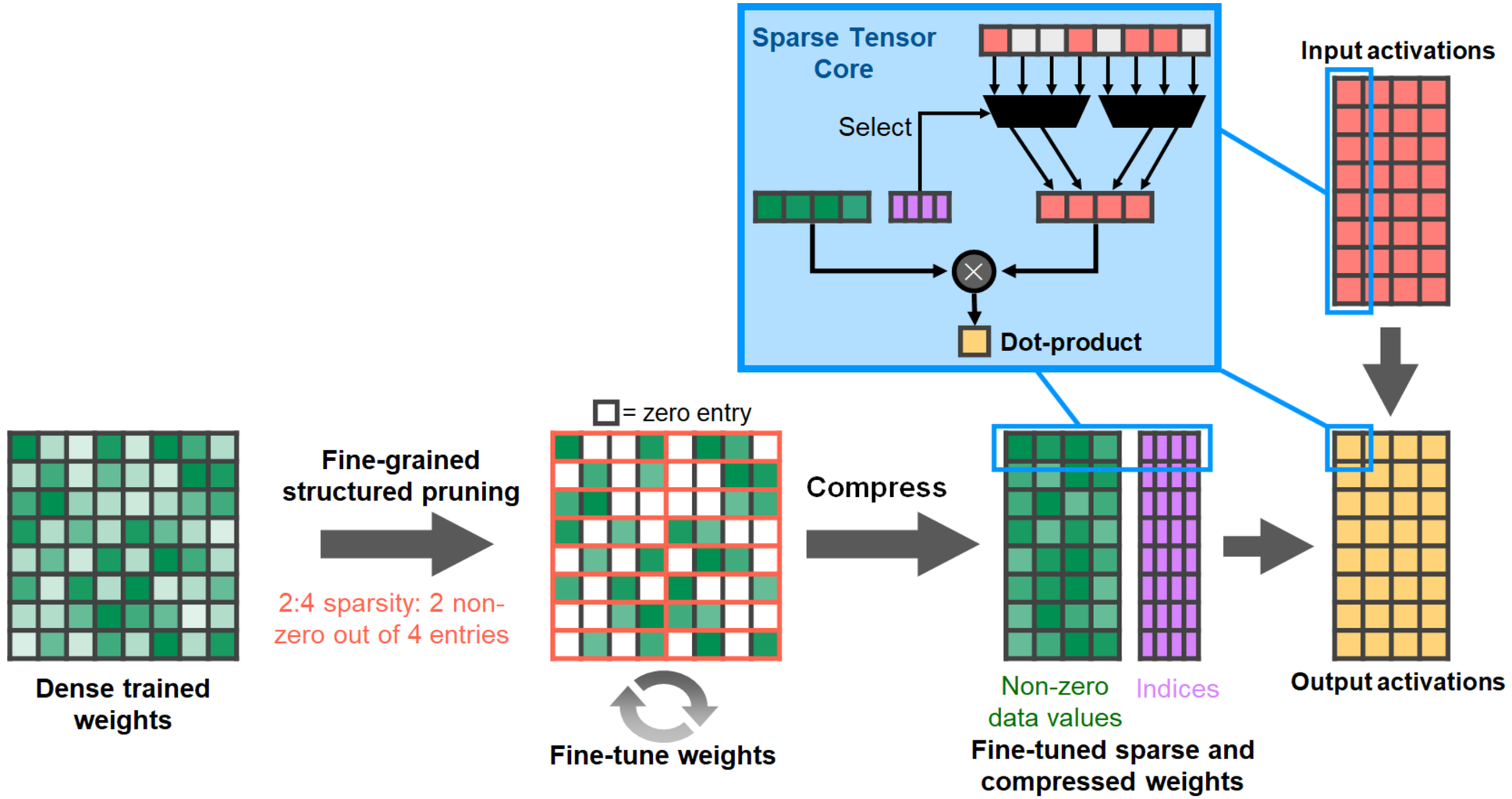

Act Queue
Act Value
Act Index
Encoded Weight
Even Ptr SRAM Bank
Odd Ptr SRAM Bank
Col Start/ End Addr
Sparse Matrix SRAM
Regs
Weight Decoder
Address Accum
Relative Index
Absolute Address
Bypass
Act SRAM
Leading NZero Detect
Dest Act Regs
Src Act Regs
ReLU
Pointer Read
Sparse Matrix Access
Arithmetic Unit
Act R/W

SCNN


F
I
Coordinate Computation
F*I
OARAM (sparse)
indices
Weight FIFO (sparse)
indices
F
IARAM (sparse)
indices
I
F*I
F*I—A arbitrated xbar
Buffer bank
Buffer bank
PPU:
Halos
ReLU
Compress
Neighbors
FxI multiplier array
A accumulator buffers


1.2
1
0.8
0.6
0.4
0.2
0
Energy
0.1/0.1
0.2/0.2
0.3/0.3
0.4/0.4
0.5/0.5
0.6/0.6
0.7/0.7
0.8/0.8
0.9/0.9
1.0/1.0
Weight / Activation Density
DCNN
DCNN-opt
SCNN

Structured Sparsity



NVIDIA A100 Tensor Core GPU Architecture whitepaper

Conclusion

Conclusion

$$E = \frac{1}{2}CV^2$$

- V^2 – Reduce V until it gets too slow (~0.5V)
- C – Communication (100fJ/b-mm), Memory (50fJ/b), Operations (Add - 1fJ/b)
- Do
- Less – overhead, communication
- Use small numbers – scale, clip, log rep
- Locally – message-driven, careful placement, NUMA
- Combinationally – flop ~= add
- Sparsely - but beware the overhead

- There are several orders of magnitude left, but it’s getting harder