Power-Efficient Neural Networks Using Low-Precision Data Types and Quantization

Tutorial at CVPR 2025, Nashville, USA, on June 12.

For material, see

https://power-efficient-nn.github.io/

Schedule

1:00 pm - 1:10 pm: Opening remarks

1:10 pm - 2:10 pm: **Session 1**

- Title: **Low-precision data types and computation**
- Speaker: Thomas Pfeil (Recogni)

2:10 pm - 3:10 pm: **Session 2**

- Title: **Quantization algorithms fundamentals**
- Speaker: Markus Nagel (Qualcomm)

3:10 pm - 3:30 pm: Coffee break ☕

3:30 pm - 4:30 pm: **Session 3**

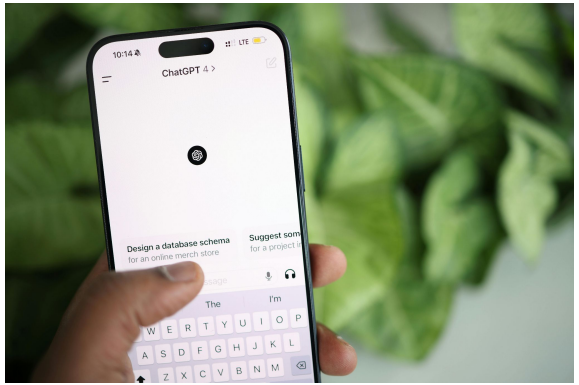
- Title: **Advanced LLM quantization methods**
- Speaker: Tijmen Blankevoort (Meta)

For material, see

https://
power-efficient-nn
.github.io/

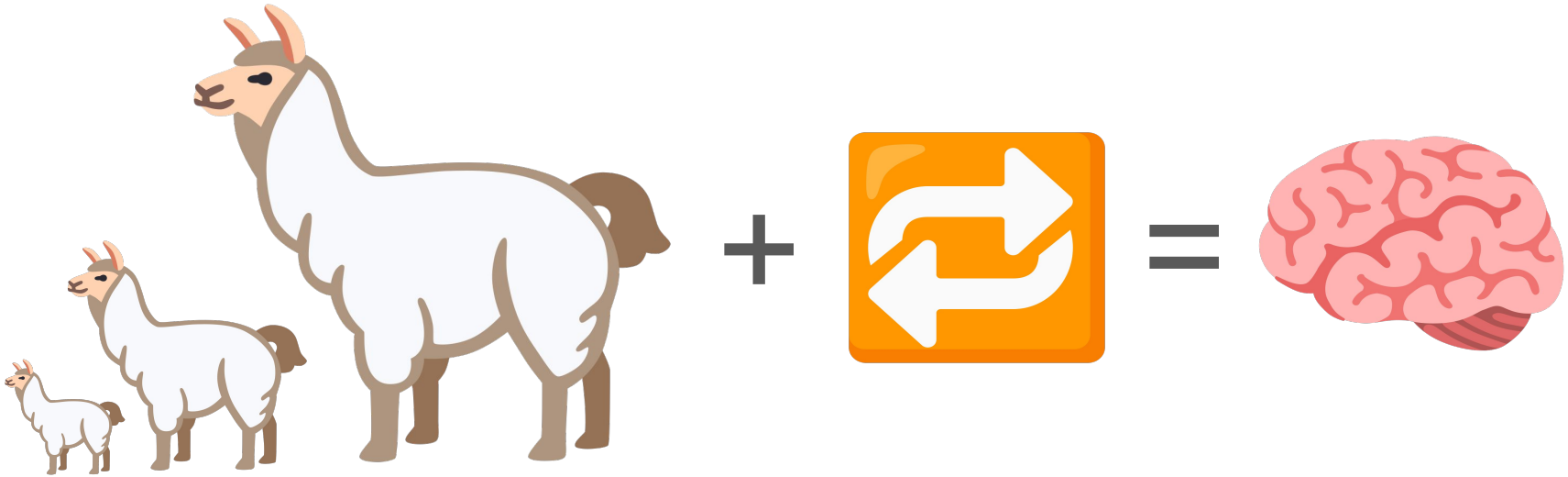
Power-efficient computing enables new applications

https://commons.wikimedia.org/wiki/File:Waymo_self-driving_car_front_view.gk.jpg

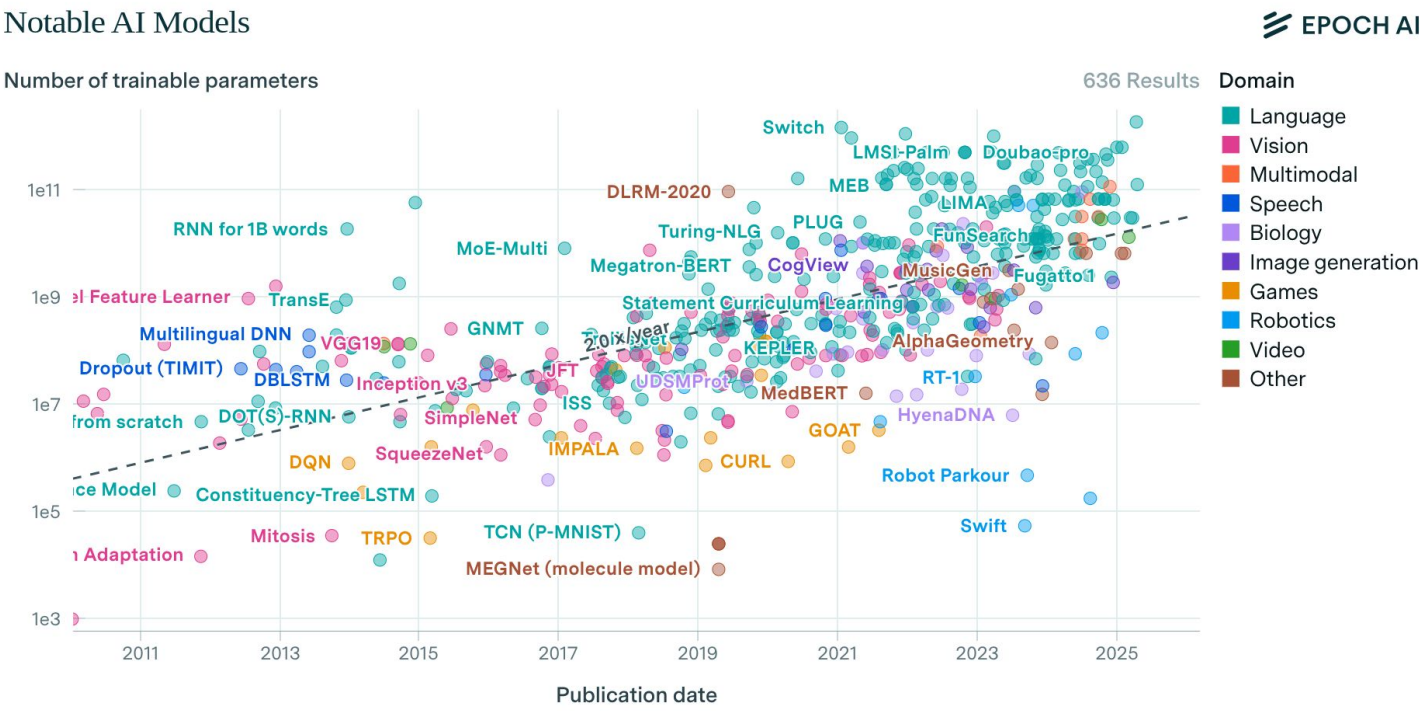


https://www.pexels.com/photo/close-up-of-a-person-holding-a-smartphone-displaying-chatgpt-20870795/

Go bigger, iterate often, get smarter

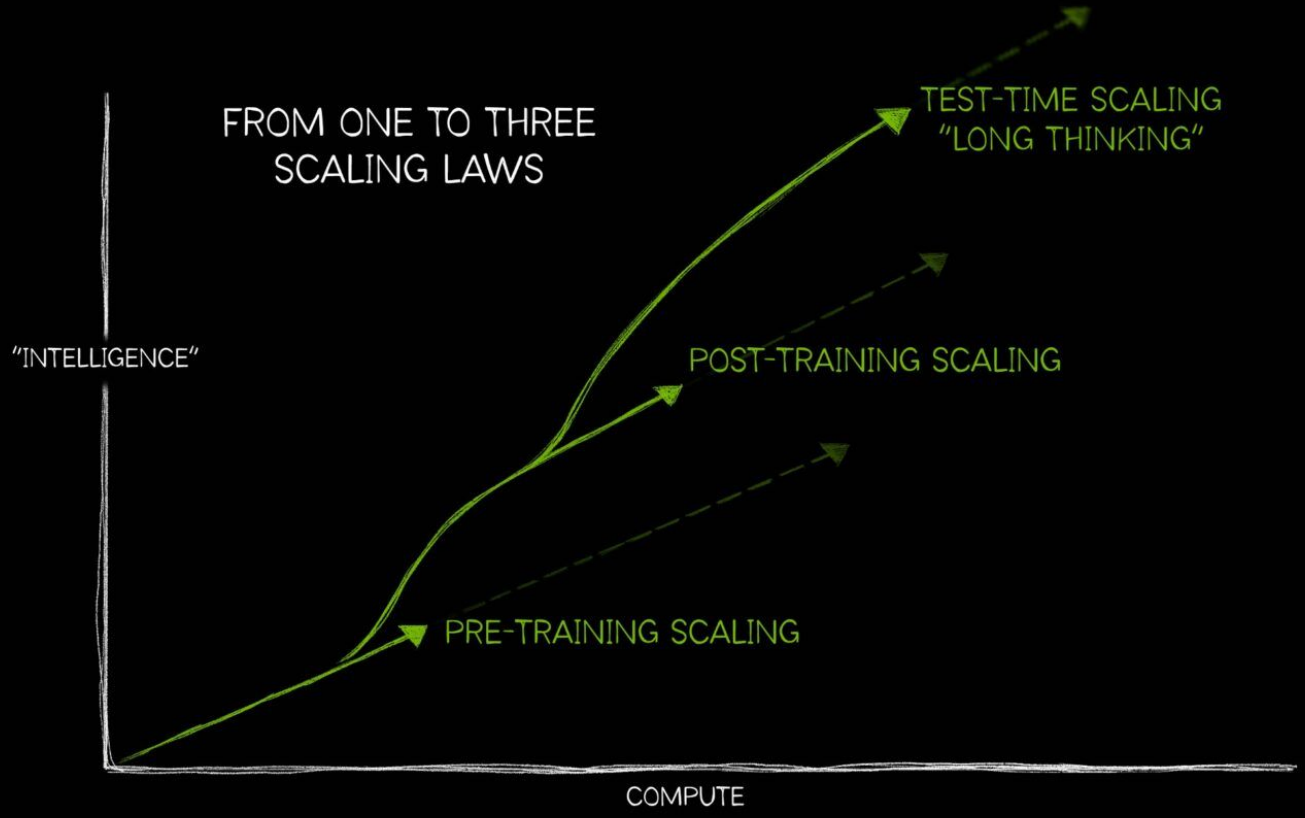


The need for storage



epoch.ai

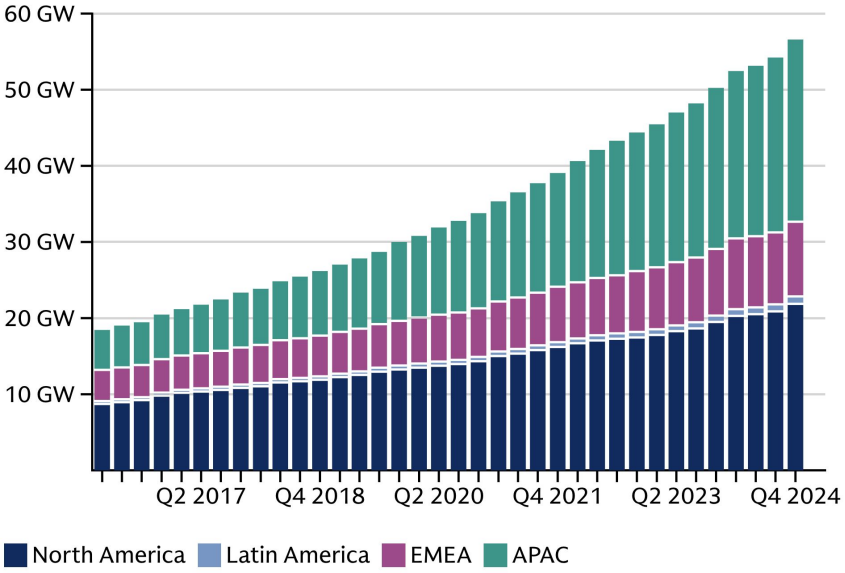
The need for compute



https://www.nvidia.com/en-us/on-demand/session/ces25-keynote/

The demand for power

Historical data center supply by region



Source: 451 Research, Goldman Sachs Research

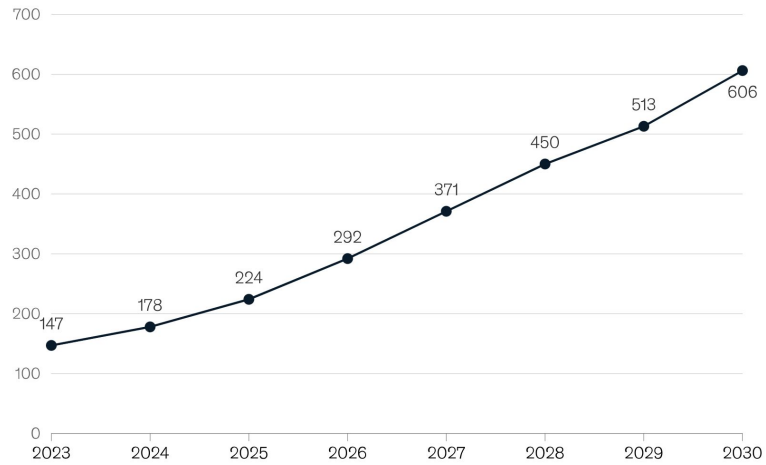
Goldman Sachs

https://www.goldmansachs.com/insights/articles/ai-to-drive-165-increase-in-data-center-power-demand-by-2030

Demand for power for data centers is expected to rise significantly in the United States.

Terawatt-hours (TWh) of electricity demand, medium scenario

US data center energy consumption, TWh



	2023	2024	2025	2026	2027	2028	2029	2030
Share of total US power demand, %	3.7	4.3	5.2	6.5	8.0	9.3	10.3	11.7

https://www.mckinsey.com/featured-insights/sustainable-inclusive-growth/charts/ais-power-binge

The limit in power availability

Explained: Generative AI's environmental impact

Rapid development and deployment of powerful generative AI models comes with environmental consequences, including increased electricity demand and water consumption.

Adam Zewe | MIT News
January 17, 2025

https://news.mit.edu/2025/explained-generative-ai-environmental-impact-0117

SUNDAY MORNING

Big Tech's big bet on nuclear power to fuel artificial intelligence

SUNDAY MORNING

By David Pogue
Updated on: March 9, 2025 / 4:47 PM EDT / CBS News

https://www.cbsnews.com/news/big-techs-big-bet-on-nuclear-power-to-fuel-artificial-intelligence/

MAY 20, 2025

Will we have enough natural gas turbines to power AI data centers?

Electricity demand is surging in the U.S., but there's years-long wait on the turbines that are a critical component of natural gas power plants.

Technology | by Elizabeth Trovall

https://www.marketplace.org/story/2025/05/20/turbine-shortage-slows-new-natural-gas-plant-construction

Part 1: Low-precision data types and computation

📉 Less bits. 🪵 Log math. 📈 More compute.

Thomas Pfeil
PhD Physics, ex-IBM and Bosch Research
Lead of hardware-algorithm co-design at recogni

Methods for network compression

- Quantization

- Pruning / Sparsity
- Low-rank approximation
- Knowledge distillation
- Neural architecture search

Getting more compute for your buck

1. Who executed a deep neural network?
2. Who executed a NN with data types <= INT16/FP16?
3. Who executed a NN with data types <= INT8/FP8?
4. Who executed a NN with data types < INT8/FP8?
5. Who quantized NN themselves?
6. Who wrote custom quantizers?
7. Who modified the definition of basic operations like multiplication or addition?

Dot products, the workhorse of AI

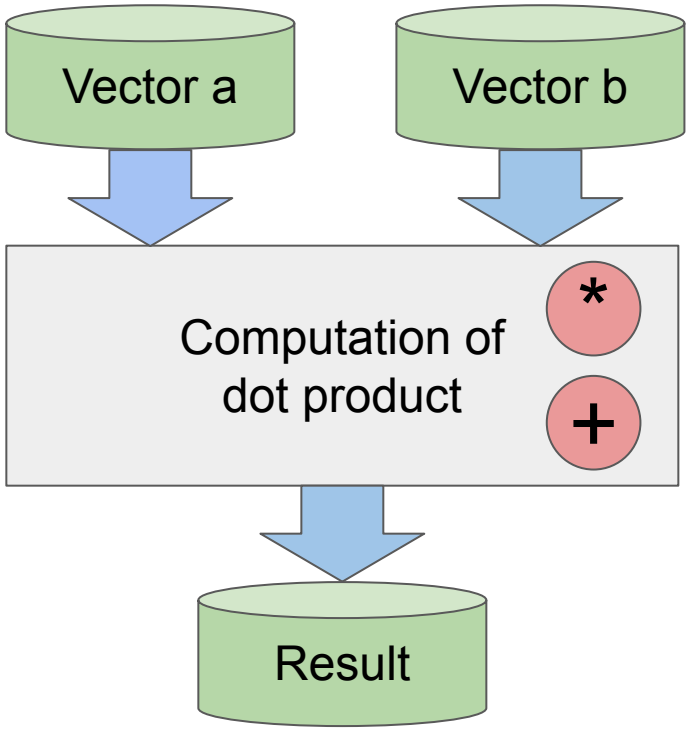
- As used in matrix multiplications and normalization layers

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\hat{x}_i = \frac{x_i}{\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}}, \quad y_i = \gamma \hat{x}_i + \beta$$

- Applies to both training and inference
  - Training has different requirements for dynamic range and precision
- Different “importance” of parameters and activations

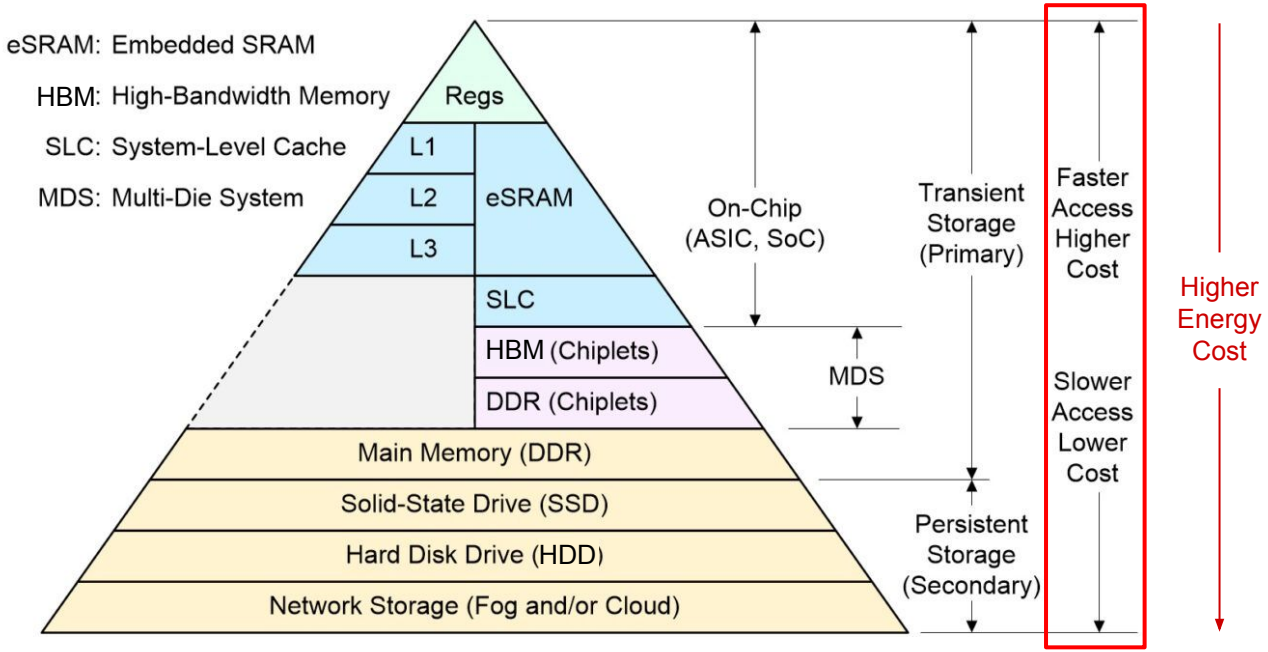
The cost of dot products



Legend:

- Storage
- Bandwidth
- Computation

The challenge to store and move data



https://www.eejournal.com/article/new-memory-architectures-for-socs-and-multi-die-systems/

Best practices:

- Keep data local
- Re-use local data
- Use hardware with a lot of fast memory
- Use fastest memory for most frequently used data

Example: Linear layer

- Keep outputs local
- Store Parameters in SRAM and/or HBM

Access to DDR requires 100x energy compared to SRAM

Reduction of storage

Quantization to low-precision data types:

Industry standard

FP32

Standard for mixed precision training and inference

FP16

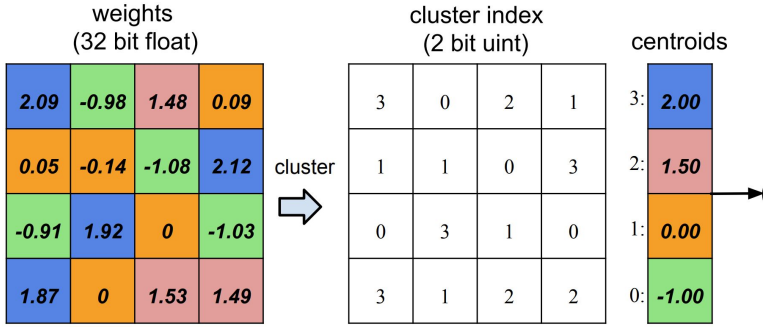
Emerging standard

FP8

Experimental

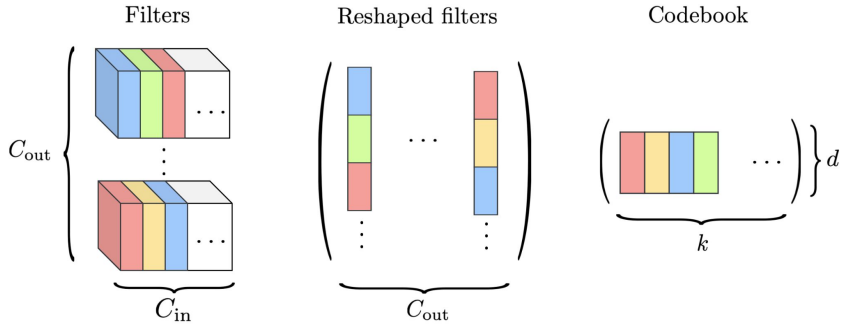
FP4

Codebook quantization / Palettization:



Han, S, Huizi M, and William J D. "Deep compression." *ICLR* (2015).

Vector quantization:



Stock, P, et al. "And the bit goes down." *ICLR* (2019).

The cost of arithmetic operations

Energy cost in pJ:

	Add		Mul	
# bits	**INT**	**FP**	**INT**	**FP**
8	0.03		0.2	
16	0.05	0.4		1.1
32	0.1	0.9	3.1	3.7

32-bit SRAM Read (8kB): 5.0 pJ
32-bit DRAM Read: 640 pJ

Horowitz, M. "1.1 computing's energy problem." *ISSCC* (2014).

On chip level:

- Multiplication is more costly than addition
- Multiplication scales worse than addition (quadratically instead of linearly)
- Access to large memory is costly
- Cost for memory access scales linearly with bit width

On system level:

- Low precision data types and compute enable more FLOPs per chip
- More FLOPs per chip reduce interconnections between chips

Multiplierless dot product using log math

Log math:

Multiplications in linear space are additions in logarithmic space:

$$\begin{aligned} a \cdot b &= 2^{\log_2(a)} 2^{\log_2(b)} && (1) \\ &= 2^{\log_2(a)+\log_2(b)} && (2) \end{aligned}$$

Mitchell's approximation:

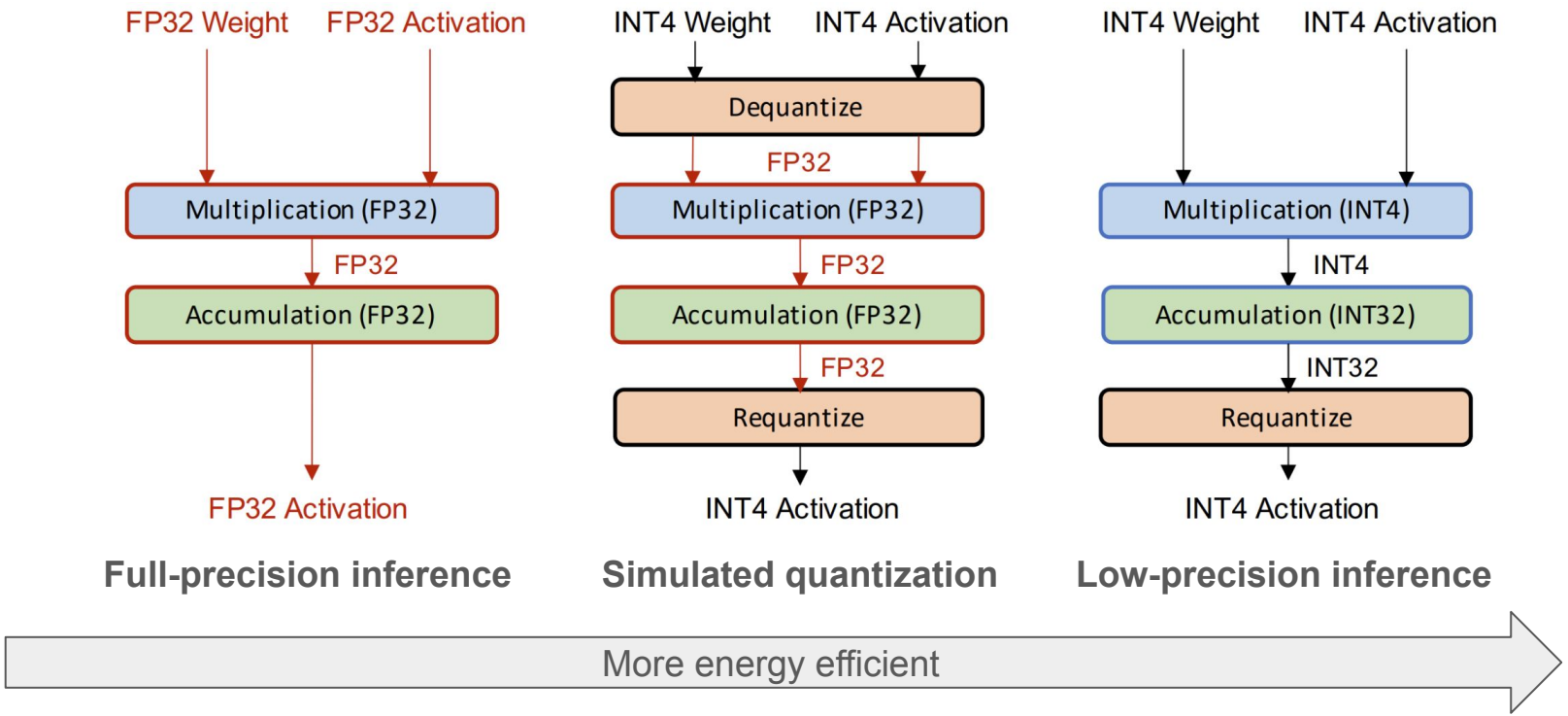
Conversion from linear to logarithmic space:

$$\begin{aligned} i.f &= \log_2(x) && (1) \\ &= \log_2(2^e(1+0.m)) && (2) \\ &= e + \log_2(1+0.m) && (3) \\ &\approx e + 0.m && (4) \\ &= e.m && (5) \end{aligned}$$

Conversion from logarithmic to linear space:

$$\begin{aligned} 2^e(1+0.m) &= 2^{i.f} && (1) \\ &= 2^i 2^{0.f} && (2) \\ &\approx 2^i(1+0.f) && (3) \end{aligned}$$

The cost of dot products



Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." Low-power computer vision. Chapman and Hall/CRC, 2022. 291-326.

Comparison of quantization methods

		Quantization to low-precision data types	Codebook quantization	Vector quantization
Energy efficiency	Storage	✅	✅✅	✅✅
	Bandwidth	✅	✅✅	✅✅
	Compute	✅✅	❌	❌
	Hardware friendliness	✅✅	✅	❌

Data types

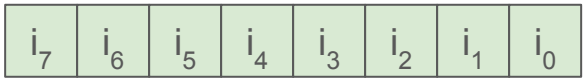
Legend:

- Integer bit
- Fraction bit
- Sign bit
- Exponent bit / Log integer bit
- Mantissa bit
- Log fraction bit

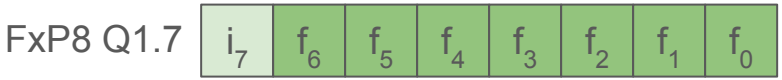
Linear data types:

$$x = s * \sum_{j=0}^{I+F-1} b_j 2^{j-F}$$

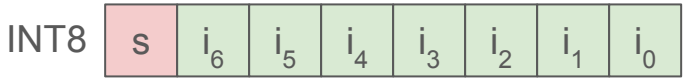
Unsigned Integer:



Fixed-point number:



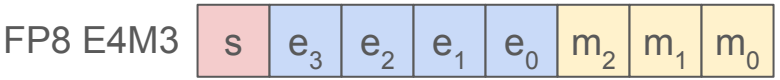
Signed Integer:



Non-linear data types:

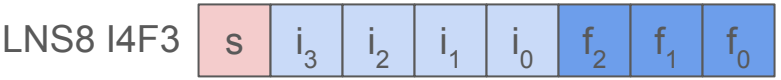
Floating-point number:

$$x = 2^e(1 + 0.m)$$



Logarithmic number system:

$$x = 2^{i.f}$$



Common data types

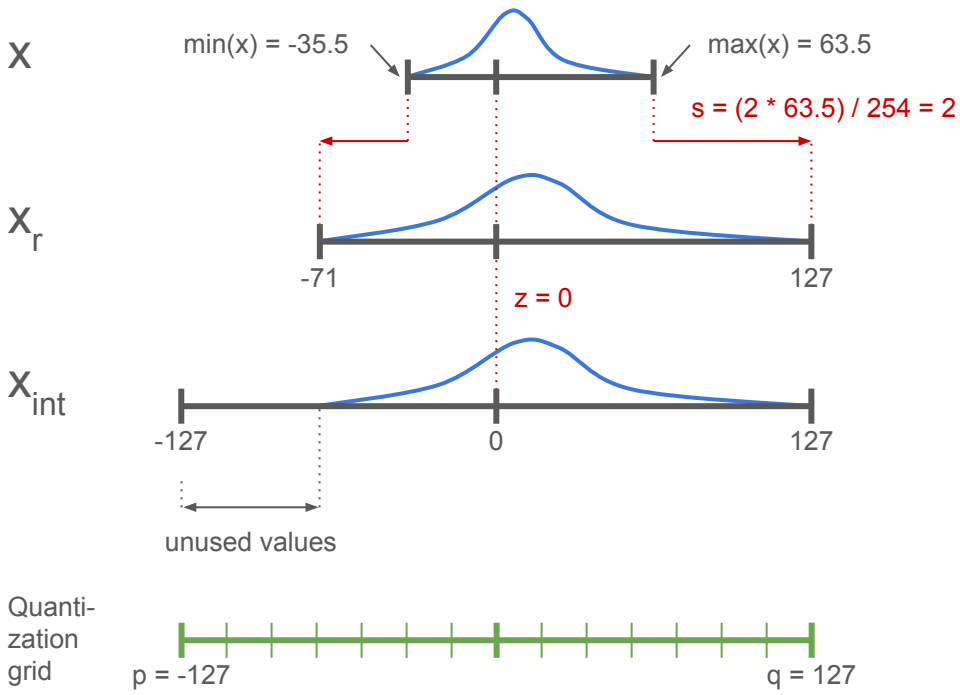
Integer data types:

	Sign bits	Integer bits
INT32	1	31
INT16	1	15
INT8	1	7
INT4	1	3

Floating-point data types:

	Sign bits	Exponent bits	Mantissa bits
FP32	1	8	23
TF32	1	8	10
BF16	1	8	7
FP16	1	5	10
FP8 (E5M2)	1	5	2
FP8 (E4M3)	1	4	3
FP4	1	2	1

Symmetric integer quantization



Quantization:

1. Scale: $x_r = \text{round}(x / s)$
2. Shift: $x_{int} = \text{clamp}(x_r + z, p, q)$
3. Reconstr.: $x_q = s(x_{int} - z) \approx x$

Typical choices for s and z:

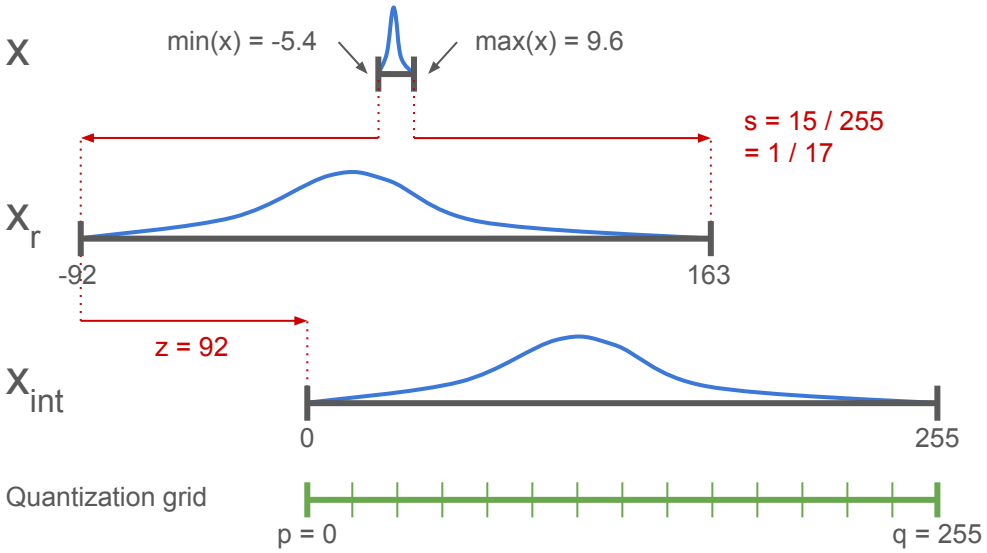
- $s = (\beta - \alpha) / (q - p)$

  with $-\alpha = \beta = \max(\text{abs}(x))$
- $z = 0$

Value range of INT with b bits:

- $p = -(2^{b-1} - 1)$; $q = 2^{b-1} - 1$

Asymmetric integer quantization



Similar methods can be applied to LNS and FP data types:

LNS: $x = 2^{s(y - b)}$ / FP: $x = 2^{s(y - b)}(1 + 0.m)^s$

Zhao, C, et al. "Insights into DeepSeek-V3" *ISCA* (2025).

Xi, H, et al. "COAT." *ICLR* (2025).

Other choices for s COVERED IN OTHER SESSIONS

Quantization:

1. Scale: $x_r = \text{round}(x / s)$
2. Shift: $x_{int} = \text{clamp}(x_r + z, p, q)$
3. Reconstr.: $x_q = s(x_{int} - z) \approx x$

Typical choices for s and z:

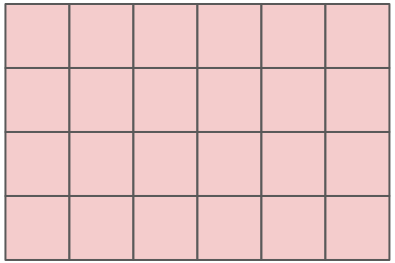
- $s = (\beta - \alpha) / (q - p)$

  with $\beta = \max(x)$; $\alpha = \min(x)$
- $z = p - \min(x_r)$

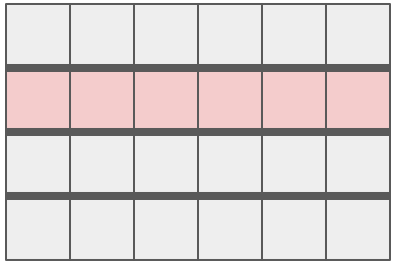
Value range of UINT with b bits:

- $p = 0$; $q = 2b - 1$

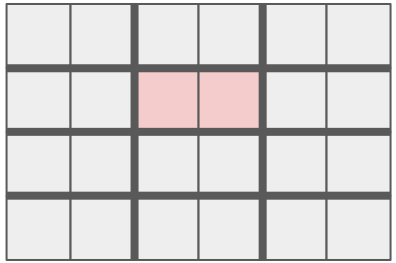
Granularity of quantization



Tensor-wise



Channel-wise



Block-wise

Block quantization

Shared parameters per block

Data type p, bit width m

Scale s

(Zero point z)

Block of scalar elements

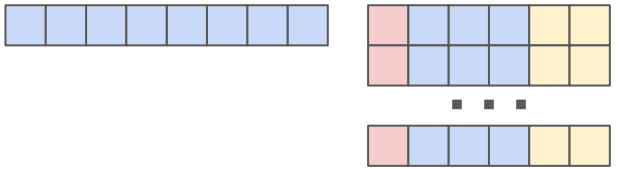
Data type q, bit width n

Elements y_i

Block size k

Examples

	MXFP6	Llama.cpp block	Llama.cpp superblock
k	32	32	$8 \cdot 32 = 256$
p, m	E8M0	INT6	FP16
q, n	FP6 (E3M2)	INT4	4.375 bits
Eff. bits per value	$(32 \cdot 6 + 8) / 32 =$ **6.25**	$(32 \cdot 4 + 2 \cdot 6) / 32 =$ **4.375**	$(256 \cdot 4.375 + 2 \cdot 16) / 256 =$ **4.5**



Block quantization

Shared parameters per block

Data type p, bit width m

Scale s

(Zero point z)

Block of scalar elements

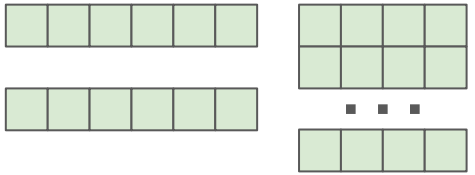
Data type q, bit width n

Elements y_i

Block size k

Examples

	MXFP6	Llama.cpp block	Llama.cpp superblock
k	32	32	$8 \cdot 32 = 256$
p, m	E8M0	INT6	FP16
q, n	FP6 (E3M2)	INT4	4.375 bits
Eff. bits per value	$(32 \cdot 6 + 8) / 32 =$ **6.25**	$(32 \cdot 4 + 2 \cdot 6) / 32 =$ **4.375**	$(256 \cdot 4.375 + 2 \cdot 16) / 256 =$ **4.5**



Block quantization

Shared parameters per block

Data type p, bit width m

Scale s

(Zero point z)

Block of scalar elements

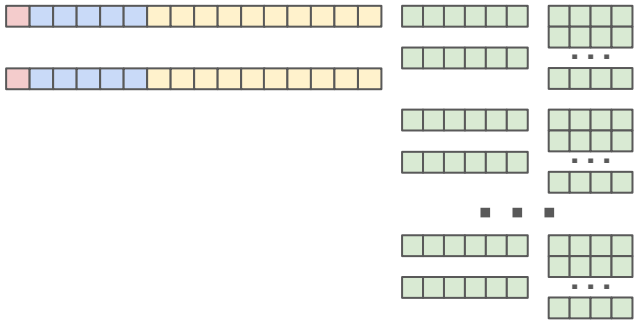
Data type q, bit width n

Elements y_i

Block size k

Examples

	MXFP6	Llama.cpp block	Llama.cpp superblock
k	32	32	$8 \cdot 32 = 256$
p, m	E8M0	INT6	FP16
q, n	FP6 (E3M2)	INT4	4.375 bits
Eff. bits per value	$(32 \cdot 6 + 8) / 32 =$ **6.25**	$(32 \cdot 4 + 2 \cdot 6) / 32 =$ **4.375**	$(256 \cdot 4.375 + 2 \cdot 16) / 256 =$ **4.5**



Dot product of MX data types

From the OCP specification of MX data types:

The dot product of two MX-compliant format vectors A: $\left\{X^{(A)}, \left[P_i^{(A)}\right]_{i=1}^{k}\right\}$ and B: $\left\{X^{(B)}, \left[P_i^{(B)}\right]_{i=1}^{k}\right\}$ of length k is a scalar number C. The following semantics *must* be minimally supported:

$$C = Dot(A,B) = X^{(A)}X^{(B)}\sum_{i=1}^{k}\left(P_i^{(A)} \times P_i^{(B)}\right)$$

Where:

- $X^{(A)}$, $X^{(B)}$ are the block scales of vectors A and B respectively.
- $P_i^{(A)}$, $P_i^{(B)}$ are the i'th element of vectors A and B respectively.

https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf

Coding session

1. Introduction of data types
2. Handling of outliers
3. Multiplierless dot products

For code, see

https://power-efficient-nn.github.io/

Data type support on cloud and edge hardware

Name	Year of introd.	FP16	BF16	INT16	FP8	INT8	FP4	INT4	MX	LNS
NVIDIA GB200	2025	✅	✅	❌	✅	✅	✅	❌	(✅)	❌
NVIDIA H200	2024	✅	✅	❌	✅	✅	❌	❌	❌	❌
Google TPU v7	Announced in 2025	?	?	?	✅	?	?	?	?	❌
Google TPU v6e	2024	❌	✅	❌	❌	✅	❌	❌	❌	❌
AMD MI325X	2024	✅	✅	❌	✅	✅	❌	❌	❌	❌
Intel Gaudi 3	2024	✅	✅	✅	✅	❌	❌	❌	❌	❌
GroqChip	2022	✅	❌	✅	❌	✅	❌	❌	❌	❌
SambaNova SN40L	2023	❌	✅	❌	❌	✅	❌	❌	❌	❌
Cerebras CS-2	2022	✅	❌	✅	❌	❌	❌	❌	❌	❌
Qualcomm SD 8	2024	✅	❌	✅	❌	✅	❌	✅	❌	❌
Apple A17	2023	✅	❌	❌	❌	✅	❌	✅	❌	❌
Mediatek MDLA 3.0	2023	✅	❌	✅	❌	✅	❌	❌	❌	❌

For dot product engines only. Only data types of bit width <=16 are listed.

Throughput for different data types

NVIDIA H100

TF32 Tensor Core²	989 TFLOPS
BFLOAT16 Tensor Core²	1,979 TFLOPS
FP16 Tensor Core²	1,979 TFLOPS
FP8 Tensor Core²	3,958 TFLOPS
INT8 Tensor Core²	3,958 TFLOPS

https://nvdam.widen.net/s/nb5zzzsjdf/hpc-datasheet-sc23-h200-datasheet-3002446

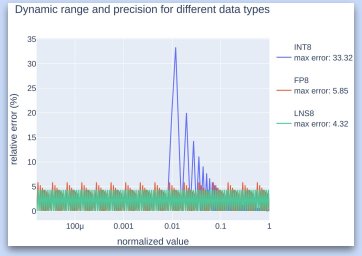
Google TPU

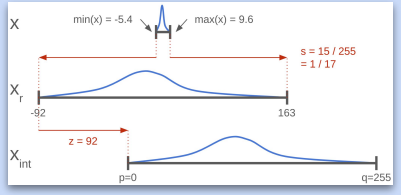
Specification	v5e	v6e
Peak compute per chip (bf16)	197 TFLOPs	918 TFLOPs
Peak compute per chip (Int8)	393 TOPs	1836 TOPs

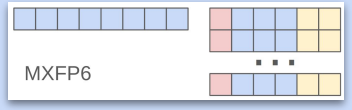
https://cloud.google.com/tpu/docs/v6e#system-architecture

Summary

Less bits.

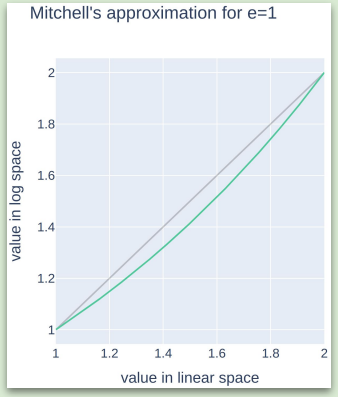




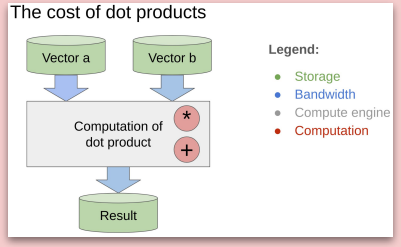


Log math.

$$a \cdot b = 2^{\log_2(a)+\log_2(b)}$$



More compute.



		Low-precision data types
Energy efficiency	Storage	✅
	Bandwidth	✅
	Compute	✅✅
	Hardware friendliness	✅✅

Backup Slides

MX support on NVIDIA Blackwell

Mode	Supported compute capabilities	Tensor values data type	Scaling factors data type	Scaling factor layout
Tensorwide scaling	8.9+	`CUDA_R_8F_E4M3` / `CUDA_R_8F_E5M2`	`CUDA_R_32F`	Scalar
Outer vector scaling	9.0	`CUDA_R_8F_E4M3` / `CUDA_R_8F_E5M2`	`CUDA_R_32F`	Vector
128-element 1D block scaling	9.0	`CUDA_R_8F_E4M3` / `CUDA_R_8F_E5M2`	`CUDA_R_32F`	Tensor
128x128-element 2D block scaling	9.0	`CUDA_R_8F_E4M3` / `CUDA_R_8F_E5M2`	`CUDA_R_32F`	Tensor
32-element 1D block scaling	10.0+	`CUDA_R_8F_E4M3` / `CUDA_R_8F_E5M2`	`CUDA_R_8F_UE8M0` [1]	Tiled tensor [3]
16-element 1D block scaling	10.0+	`CUDA_R_4F_E2M1`	`CUDA_R_8F_UE4M3` [2]	Tiled tensor [3]

https://docs.nvidia.com/cuda/cublas/index.html#d-block-scaling-factors-layout