Precision Matters: Quantization, Pruning, and Fine-Tuning for Efficient AI

Mert Pilanci

Stanford University

2026

Outline

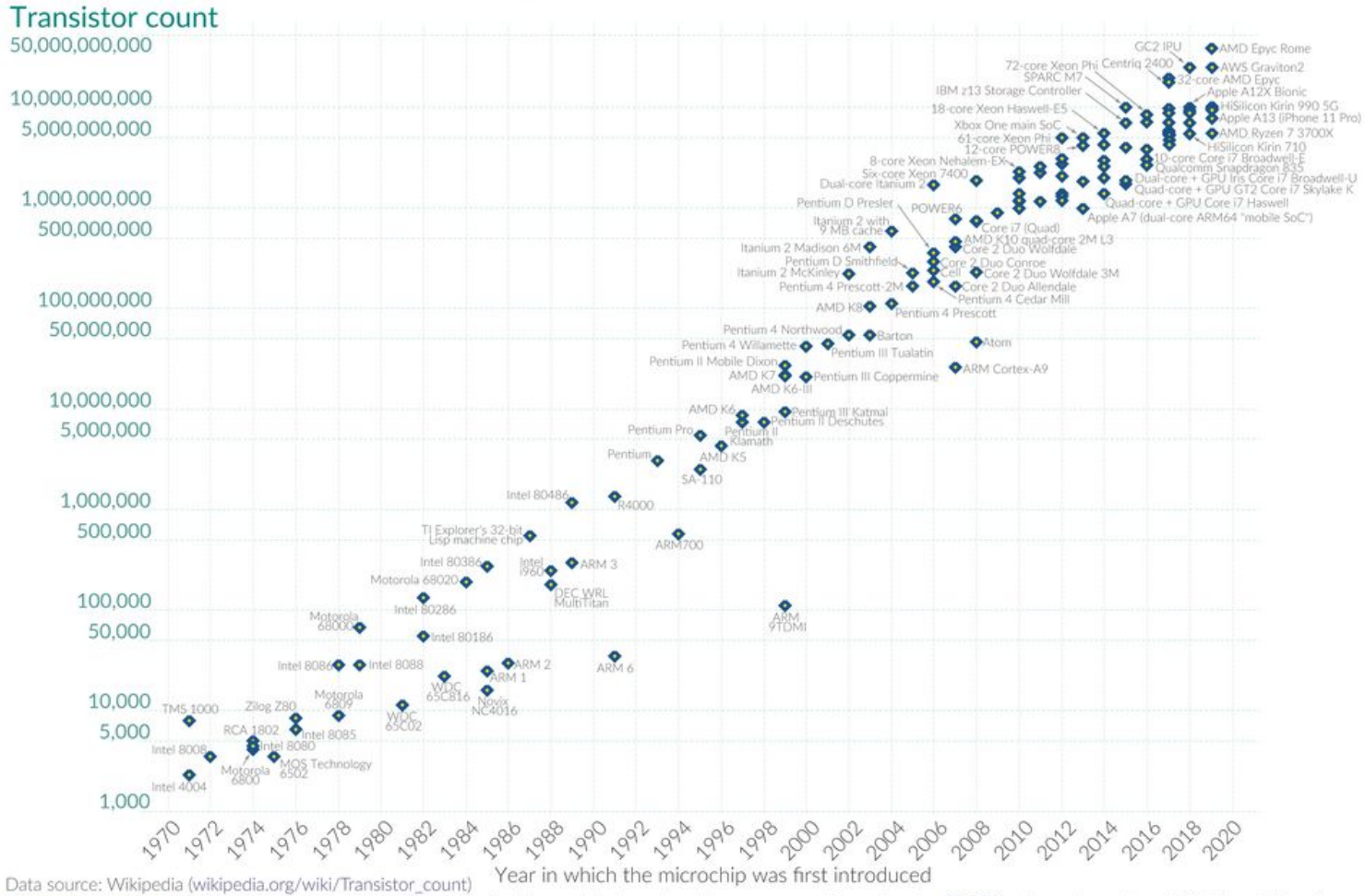
- trends in AI
- a short history of numerical precision

- current challenges
  - Machine Learning (ML)
  - Large Language Models (LLMs)
- compressing neural networks
  - quantization
  - low-rank models
  - mixed-precision
  - fine-tuning
  - insights from theory
- future work

Current Trends in AI

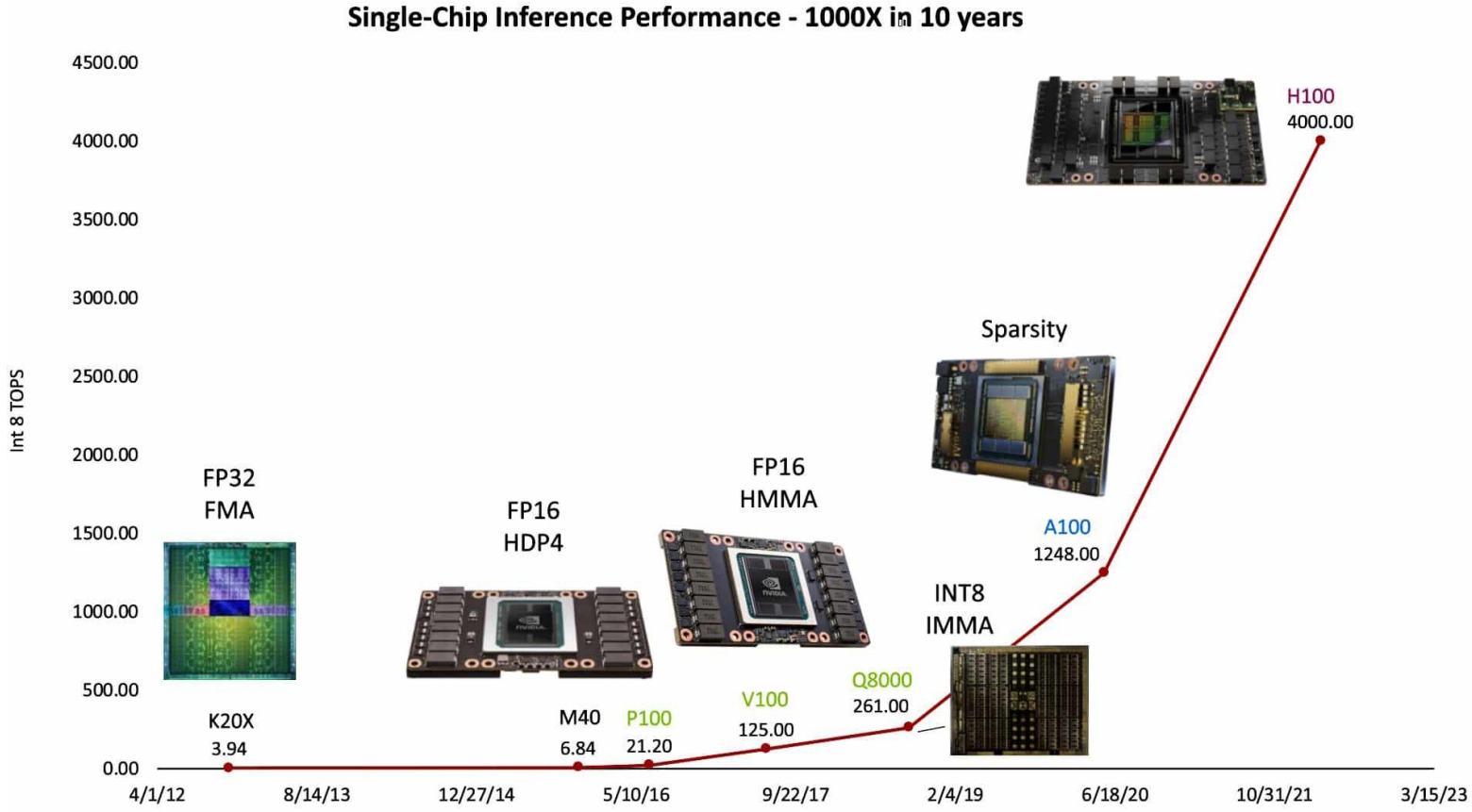
now

Moore's Law

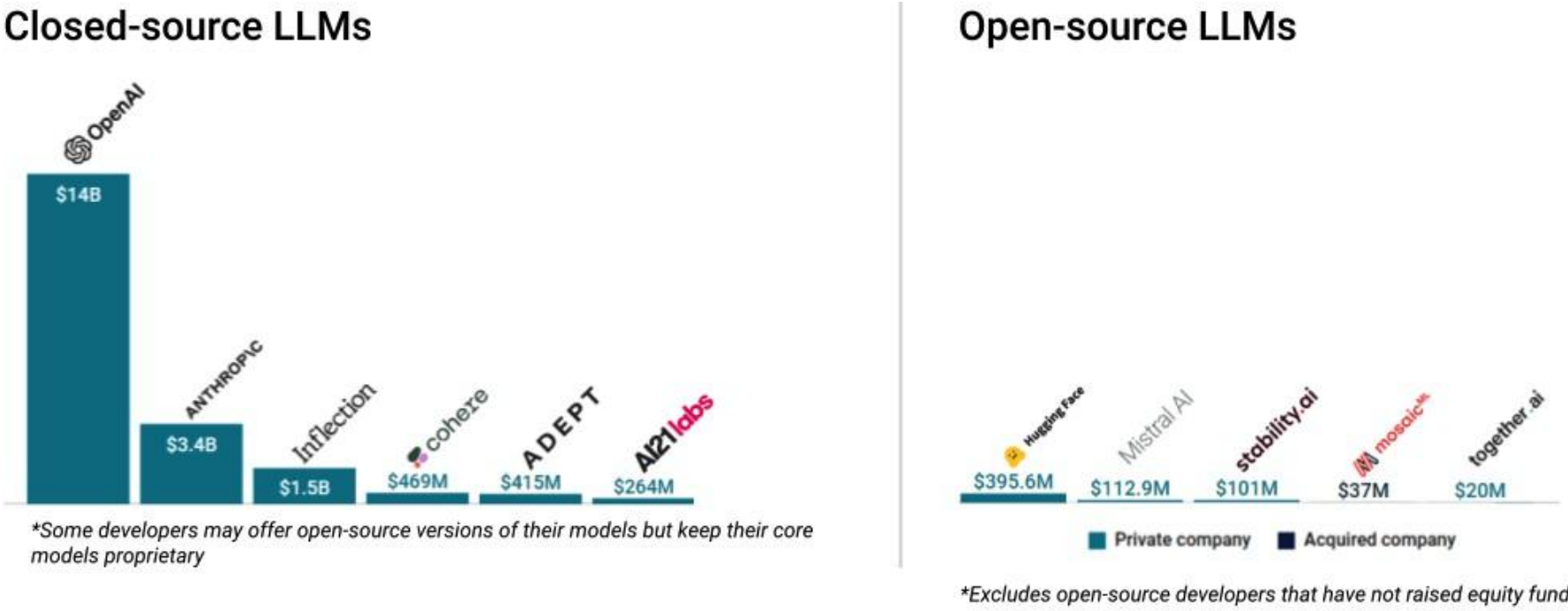


Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.

Huang's Law

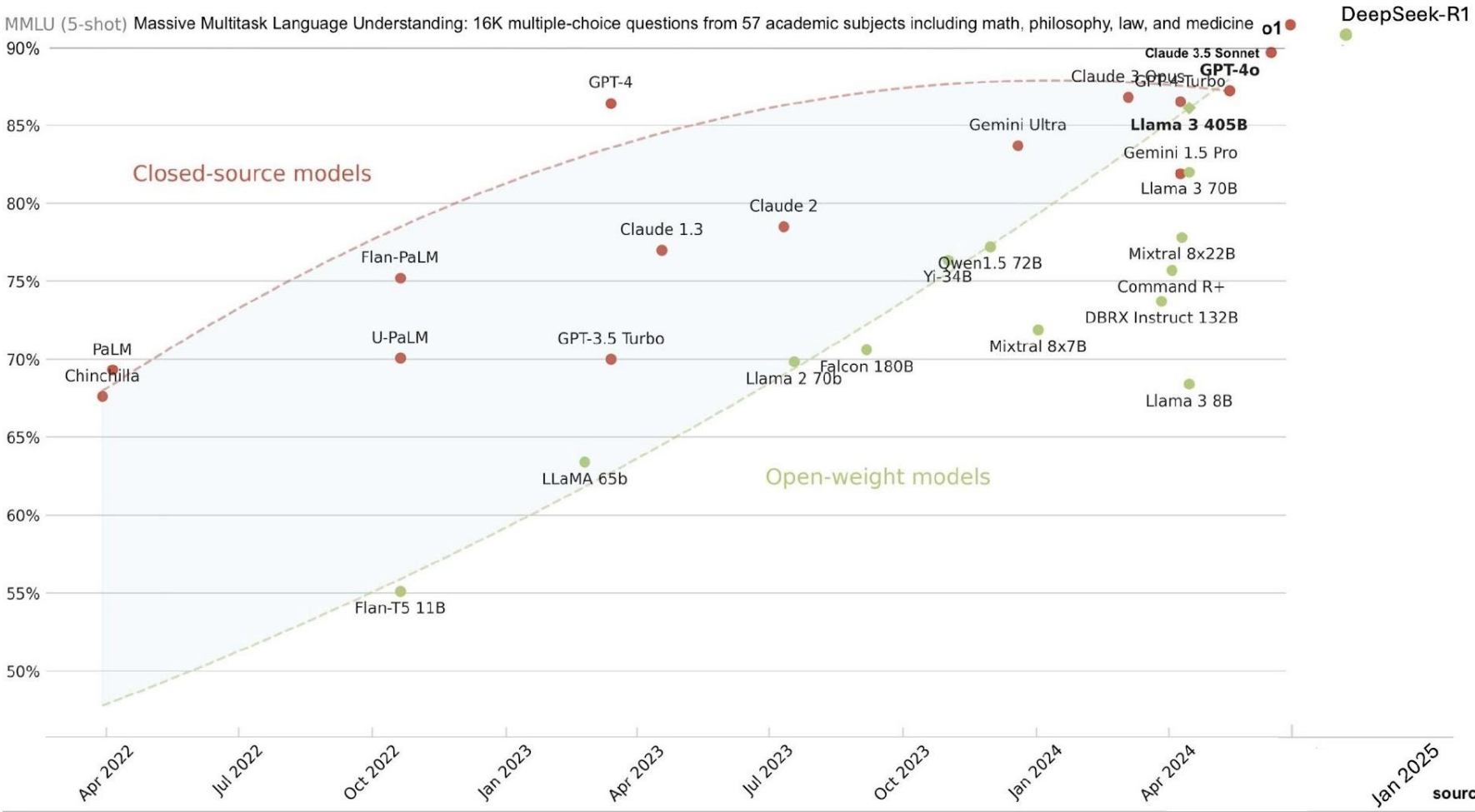

Single-Chip Inference Performance - 1000X in 10 years
Int 8 TOPS
4500.00
4000.00
3500.00
3000.00
2500.00
2000.00
1500.00
1000.00
500.00
0.00
4/1/12
8/14/13
12/27/14
5/10/16
9/22/17
2/4/19
6/18/20
10/31/21
3/15/23
FP32
FMA
FP16
HDP4
FP16
HMMA
Sparsity
INT8
IMMA
K20X
3.94
M40
6.84
P100
21.20
V100
125.00
Q8000
261.00
A100
1248.00
H100
4000.00

LLM development is split into open vs closed-source



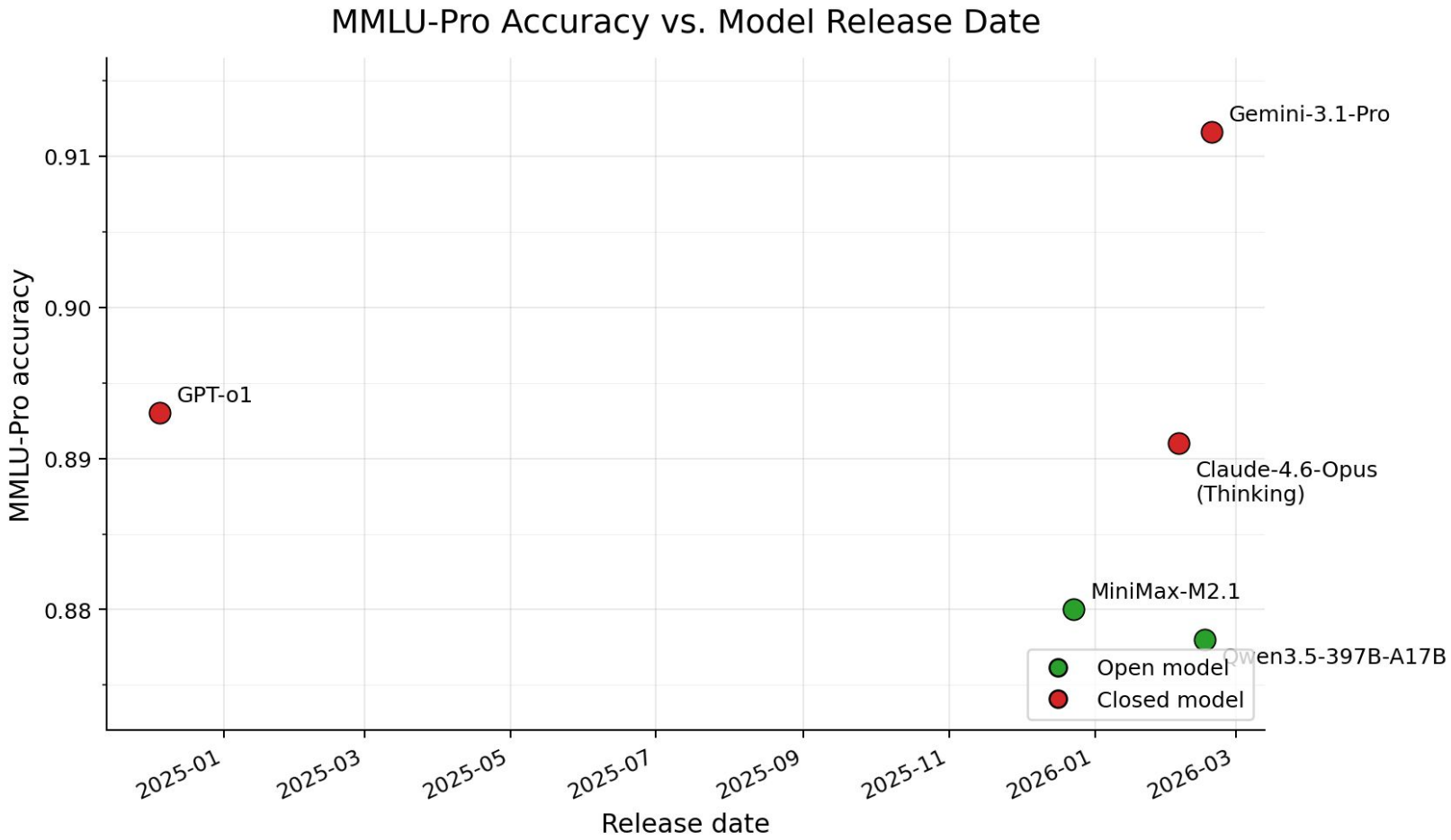
CBINSIGHTS Source: CB Insights market report: Generative AI – large language model developers

The gap between open and closed-source is narrowing



source: M. Labonne

The gap between open and closed-source is narrowing



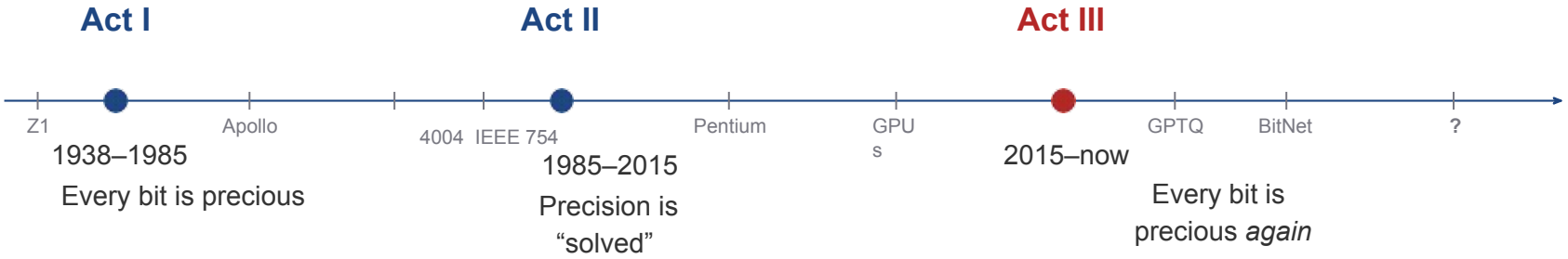
Transformer efficiency stack:

- low precision (quantizing network weights/activations)
- mixed precision
- pruning
- distillation
- fine-tuning

Benefits include:

- deployment on affordable consumer hardware accelerated inference speed
- reduced energy consumption and carbon footprint decreased memory requirements
- enabling edge deployment

A Story in Three Acts



"The irony of modern computing: we spent decades escaping low-precision arithmetic, only to return to it voluntarily."

—

ACT I

When Every Bit Was Precious

1936 – 1985

Landing on the Moon with 15 Bits (1969)

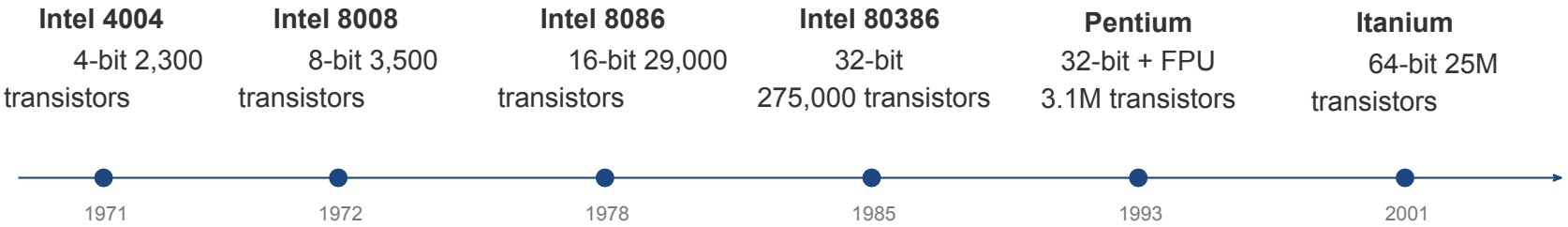
The **Apollo Guidance Computer** was a 15-bit machine.

Precision was hand-allocated:

- Angles: single precision (14 bits)
- Distances & velocity: double (28 bits)
- Elapsed time: *triple* precision

▶ Double-precision (28 data bits) was deemed “satisfactory for navigation accuracy” — MIT Instrumentation Lab, 1966.

The March of Bits: From 4 to 64



The Intel 4004 processed 4-bit words. It could represent signed integers from −8 to +7.

- Each doubling of word size opened entirely new categories of applications.

The Patriot Missile Failure (1991)

28 American soldiers killed by a fixed-point rounding error.

- Feb 25, 1991, Dhahran, Saudi Arabia — a Scud missile hits Army barracks
- The Patriot missile defense system failed to track the incoming Scud
- **Cause**: the system clock (in 1/10s of a second) was multiplied by 0.1 in a **24-bit fixed-point register**

The binary expansion of 1/10 is non-terminating:

$$\frac{1}{10} = 0.0001100110011\overline{0011}\ldots_2$$

Chopped at 24 bits, the relative error $\approx 2^{-20} \approx 0.0001\%$.

After **100 hours** of continuous operation: drift of **0.34 seconds**.

A Scud at 1,676 m/s travels **570 meters** in 0.34 seconds — far outside the tracking range gate.

▸ The corrected software arrived on February 26 — one day too late. *(GAO Report IMTEC-92-26)*

Paranoia: A Floating-Point Benchmark (1983)

William Kahan wrote a program called **“Paranoia”** to test whether your computer could do arithmetic correctly.

- Tests for: rounding errors, guard digits, normalization, overflow/underflow

▶ Published as “Paranoia: A Floating-Point Benchmark” in *Byte Magazine*, February 1985.

ACT II

The Golden Age

Someone else solved precision. Right?

The Era of “Precision is Solved” (1990s–2010s)

IEEE 754 became universal. FP32 and FP64 were everywhere.

- `float`: 32-bit, ~7 decimal digits, range ~ $10^{\pm 38}$
- `double`: 64-bit, ~16 decimal digits, range ~ $10^{\pm 308}$
- Programmers stopped thinking about precision
- Undergraduate textbooks treated floating-point as a solved problem

ACT III

Low Precision Strikes Back

Quantization returns — this time on purpose

The Memory Wall: Why We Need Quantization Again

LLMs are too large to run at full precision on available hardware.

Model	Parameters	FP16 Size	INT4 Size
LLaMA-7B	7B	14 GB	3.5 GB
LLaMA-70B	70B	140 GB	35 GB
LLaMA-405B	405B	810 GB	~200 GB
GPT-4 scale	~1.8T	~3.6 TB	~900 GB

- A single A100 GPU has 80 GB of HBM
- Reducing from 16 bits to 4 bits ⇒ 4× less memory, ~4× faster inference

▶ The same pressure that made Apollo engineers count every bit now applies to every GPU cluster running LLMs.

Example: DeepSeek and Meta open-source LLMs

inference costs

DeepSeek R1 671B (released January 20, 2025)

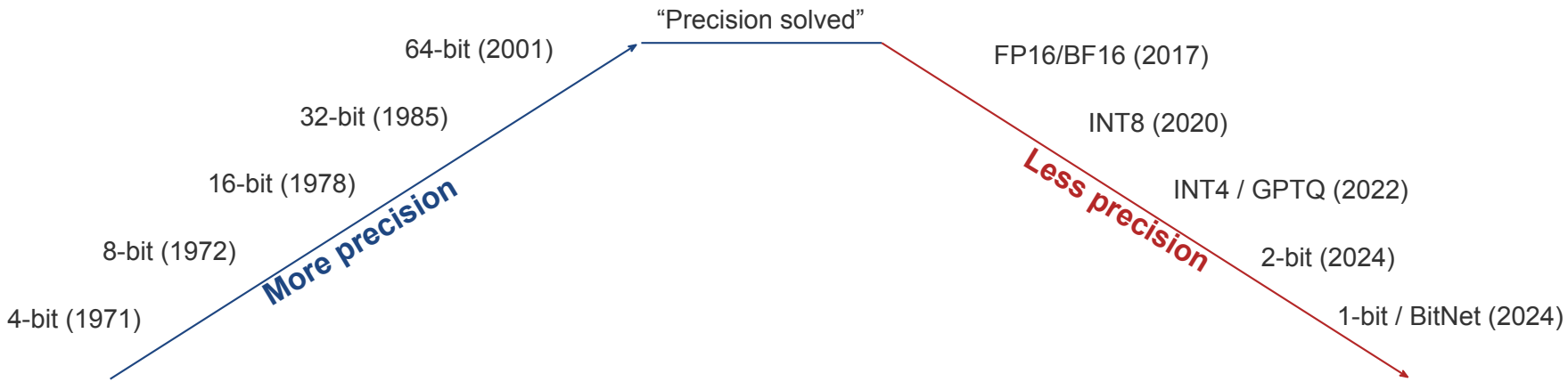
- original model (FP16): 8× H100 GPUs ($65K each)
- 2-bit quantization: single Mac Studio ($5.6K)

Llama 3 70b

- original model (FP16): 2× H100 GPUs
- 2-bit quantization: single 4090 GPU
- 2-bit quantization: single Macbook Pro M2

low-precision inference and training is more important than ever

History Repeating In Reverse



The computational history is **folding back on itself**.

- Modern 4-bit LLM quantization uses fewer bits than a computer built in a living room 87 years ago.

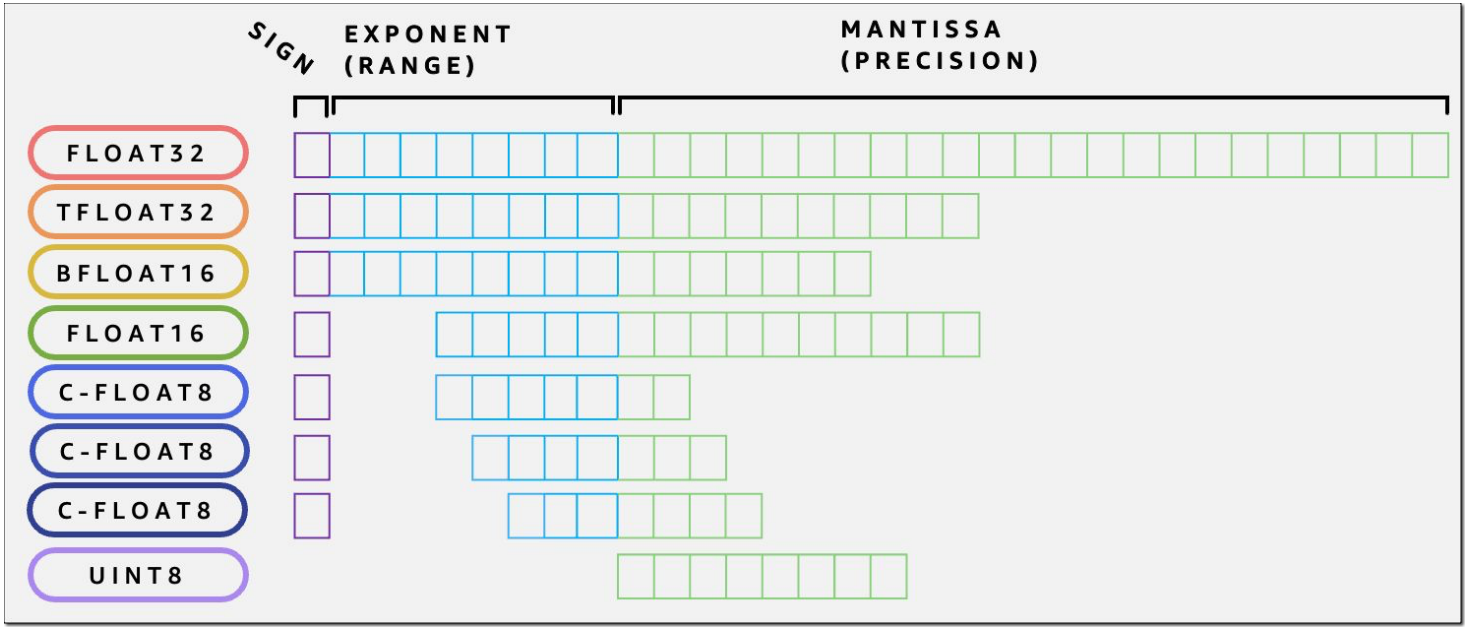
The New Zoo of Number Formats

Just as the 1970s had a chaos of formats, we now have a proliferation of low-bit representations:

Format	Bits	Type	Used In
FP32	32	IEEE 754 float	Training (legacy)
BF16	16	Brain float	Training, mixed precision
FP16	16	IEEE 754 half	Inference, fine-tuning
FP8 (E4M3/E5M2)	8	Mini-float	H100 Transformer Engine
NVFP4	4	NVIDIA format	Blackwell GPUs
INT8	8	Uniform integer	PTQ, W8A8
INT4	4	Uniform integer	GPTQ, AWQ, GGUF
NF4	4	Normal float	QLoRA
INT2	2	Binary codes	QuIP#, AQLM
1-bit	1	$\{-1, +1\}$	BitNet, OneBit

- IEEE P3109 working group (est. 2024) is standardizing sub-16-bit formats for ML

The New Zoo of Number Formats



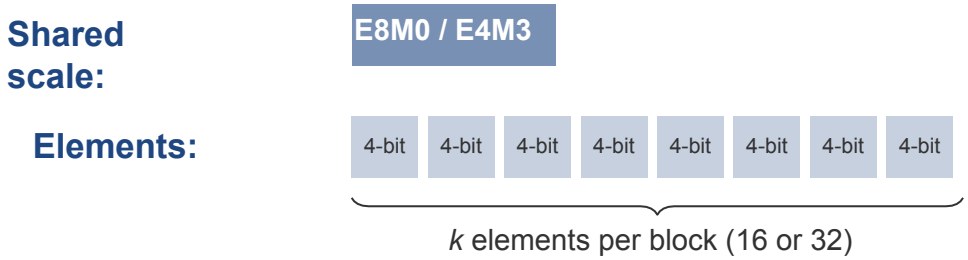
EE269 - Signal Processing and Quantization for Machine Learning

https://web.stanford.edu/class/ee269/

Discrete signals, quantization and change of basis
Quantization noise
Non-uniform quantizers, Lloyd-Max optimality and high-rate theory
Dithering and stochastic rounding
Discrete Fourier Transform (DFT), DFT based spectral descriptors
Short-Time Fourier Transform
Distance based signal classification, nearest neighbor classifier and Hilbert spaces
Continuous and Discrete Wavelet Transform
Applications of Wavelets and Short-Time Fourier Transform: Signal classification, separation and denoising
Linear systems and eigenvector decomposition
Cepstrum and Mel-frequency Cepstral Coefficients (MFCC)
Stationary signals, autocorrelation, linear and quadratic discriminant analysis
Fisher's linear discriminant, simultaneous diagonalization
Multi-class discriminant, separating hyperplanes and Support Vector Machine (SVM)
Nonlinear features, kernels and kernel machines
Least-squares regression and autoregressive models
Neural networks, adaptive filters, Least Mean Squares algorithm
Deep learning, convolutional networks and spectrograms
Attention
Transformers for signal prediction
Diffusion models for waveform and image generation

Block Floating Point: An Old Idea Returns

Block floating point (BFP): a group of values shares a single exponent (scale factor).



Historical lineage:

- **1960s**: BFP used in DSP and fixed-point processors to approximate floating-point arithmetic. J. H. Wilkinson extensively studied BFP algorithms
- **1990s–2000s**: Texas Instruments TMS320 DSPs used BFP for FFT implementations
- **2020**: Microsoft Floating Point (MSFP) — shared exponent for DNN inference
- **2023**: OCP Microscaling (MX) Specification v1.0 — industry standard
- **2024–25**: NVIDIA NVFP4 on Blackwell — hardware-native block scaling

▶ BFP was invented to give fixed-point processors floating-point capabilities. Now it returns to give tiny formats sufficient dynamic range.

The MX Standard: Industry Convergence (2023)

In September 2023, seven companies standardized microscaling formats through the Open Compute Project:

AMD · Arm · Intel · Meta · Microsoft · NVIDIA · Qualcomm

Format	Block (k)	Scale	Element	Eff. bits/elem
MXFP8	32	E8M0 (8b)	E4M3 / E5M2 (8b)	8.25
MXFP6	32	E8M0 (8b)	E2M3 / E3M2 (6b)	6.25
MXFP4	32	E8M0 (8b)	E2M1 (4b)	4.25
MXINT8	32	E8M0 (8b)	INT8 (8b)	8.25

Inside E2M1: The 4-bit Floating-Point Codebook

The E2M1 format (1 sign + 2 exponent + 1 mantissa) has only **16 representable values**:

Bits	S	E	Value	Bits	S	E	Value
0000	+	00	0	1000	−	00	−0
0001	+	00	0.5	1001	−	00	−0.5
0010	+	01	1.0	1010	−	01	−1.0
0011	+	01	1.5	1011	−	01	−1.5
0100	+	10	2.0	1100	−	10	−2.0
0101	+	10	3.0	1101	−	10	−3.0
0110	+	11	4.0	1110	−	11	−4.0
0111	+	11	6.0	1111	−	11	−6.0

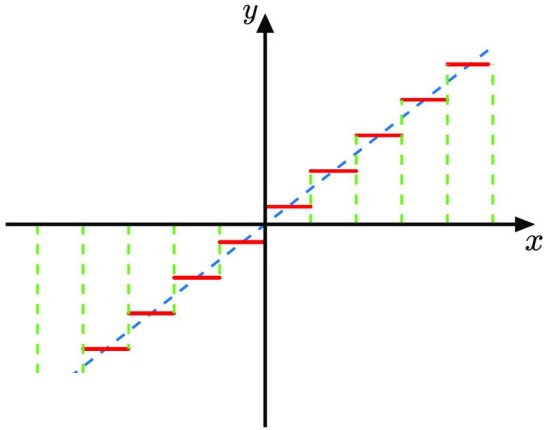
Range: only [−6, +6]. The block scale factor restores dynamic range.

Key insight: the non-uniform spacing (0, 0.5, 1, 1.5, 2, 3, 4, 6) is denser near zero — matching the typical bell-shaped weight distribution of neural networks.

Quantization: The Core Problem

Given a real-valued weight $w \in \mathrm{R}$, map it to a discrete set Q with 2^b levels:

$$Q(w) = \arg\min_{q \in \mathcal{Q}} |w - q|$$



Number of bits: B, Range boundaries: $x_{\min}, x_{\max}$

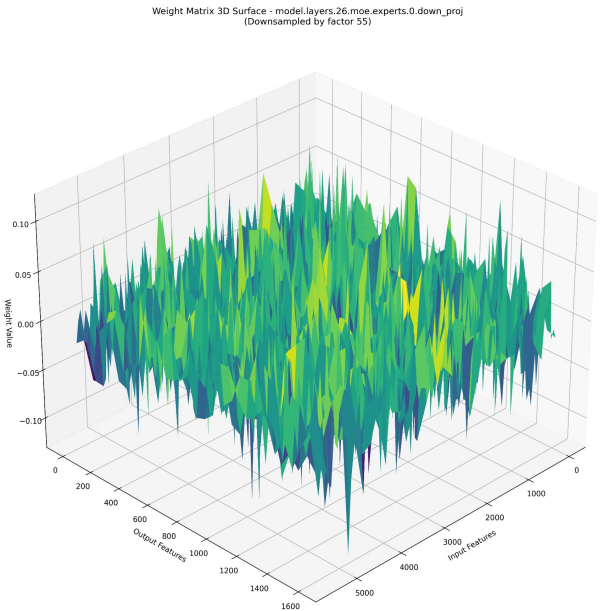
Quantization of a value x to 2^B levels:

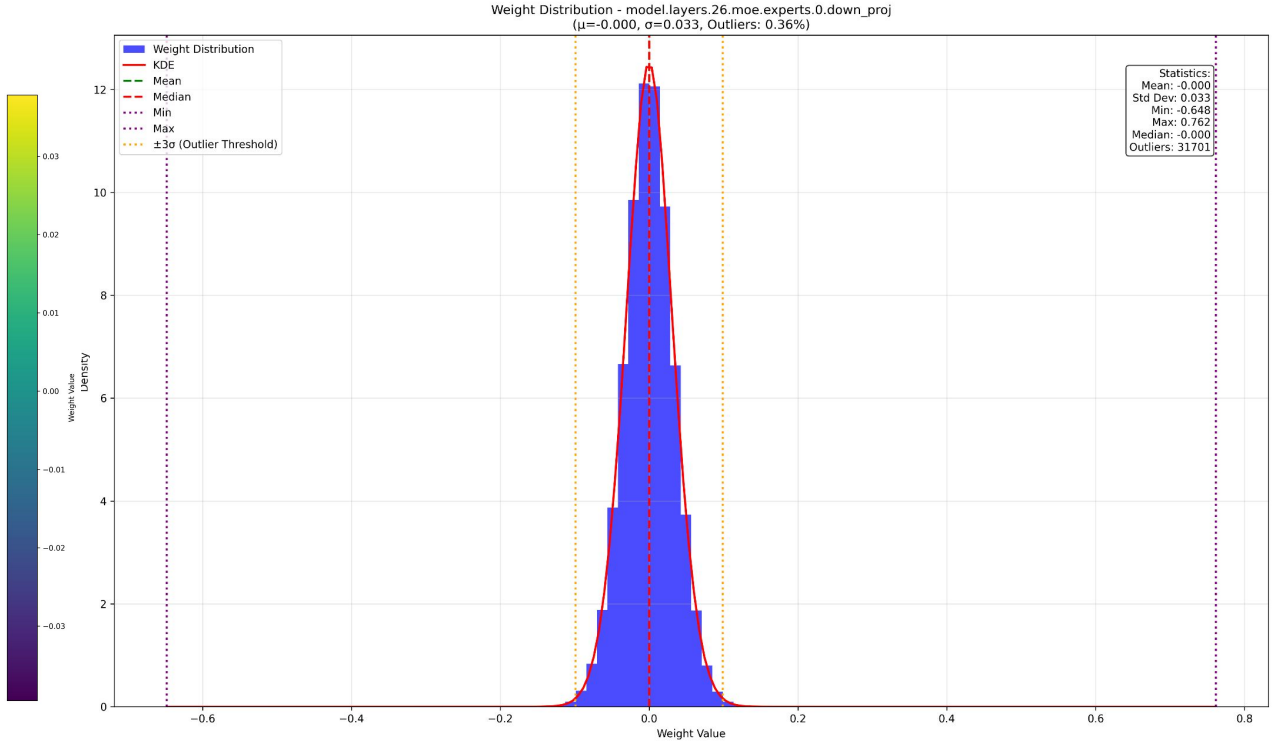
$$x_q = \text{round}\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \times (2^B - 1)\right)$$

Reconstruction from quantized value (and scale factor + zero-point)

$$x \approx x_{\text{rec}} = x_{\min} + \frac{x_q}{2^B - 1}(x_{\max} - x_{\min})$$

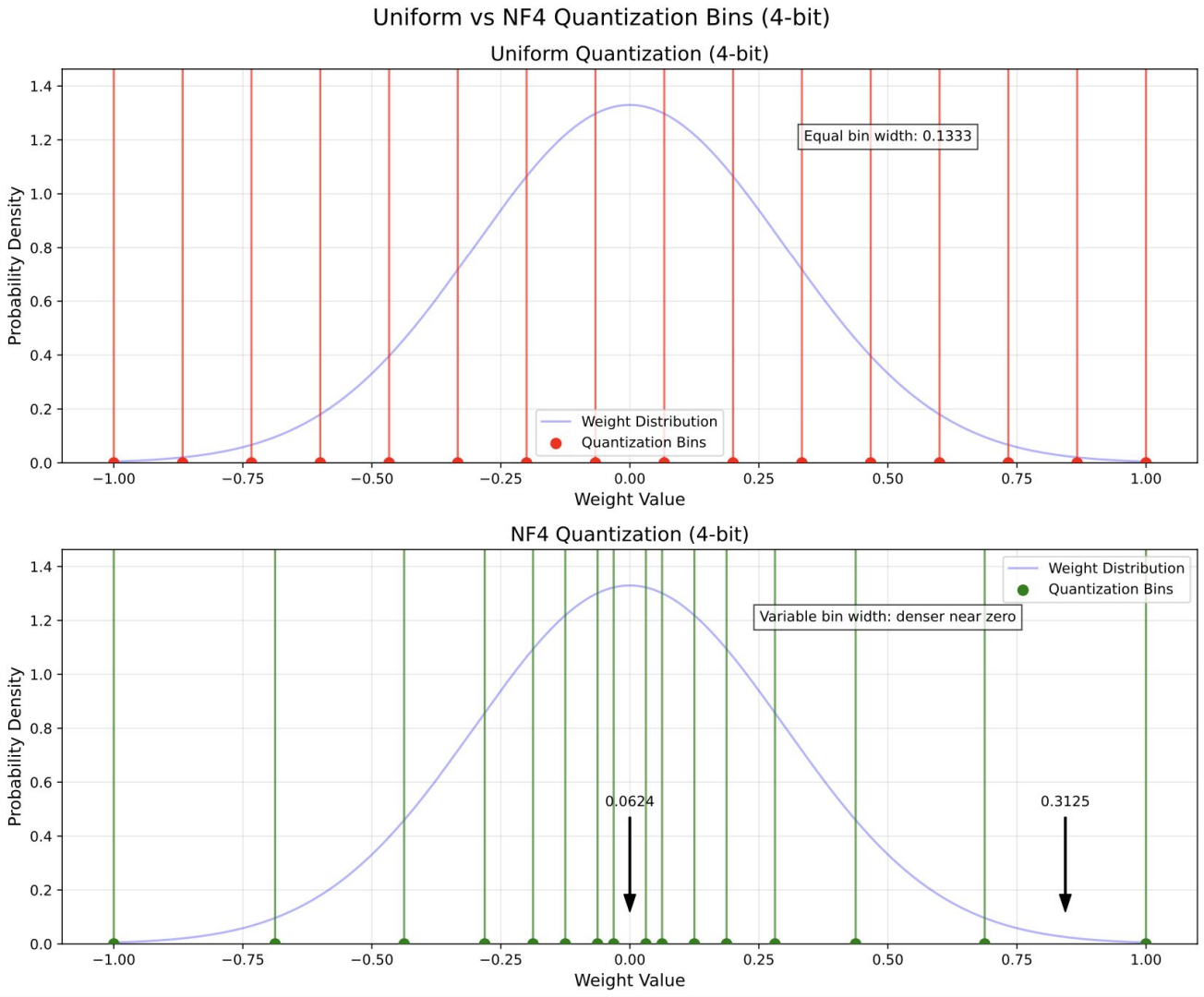
Distribution of LLM Weights

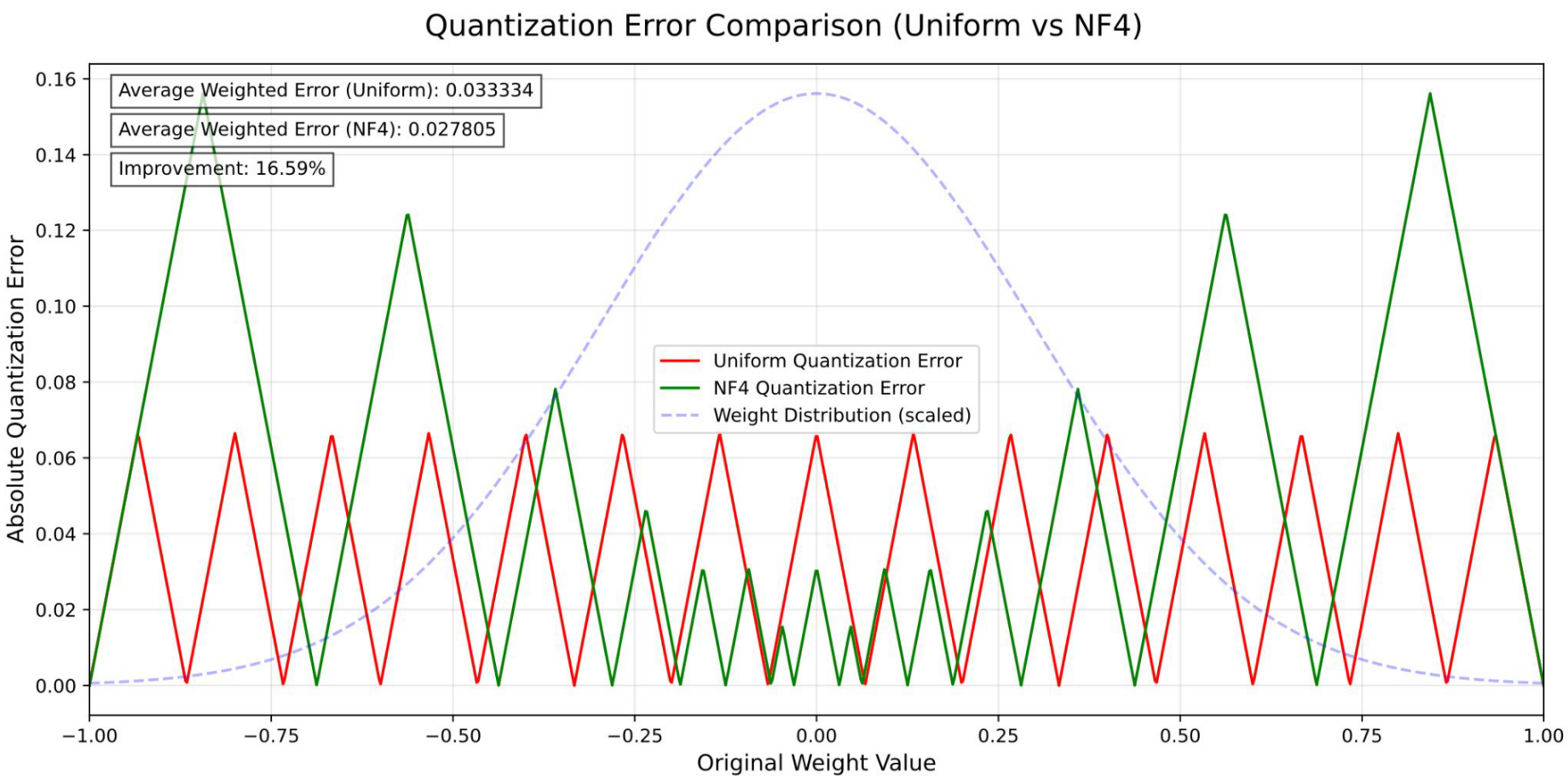




NormalFloat NF4 (Dettmers et al. 2023)

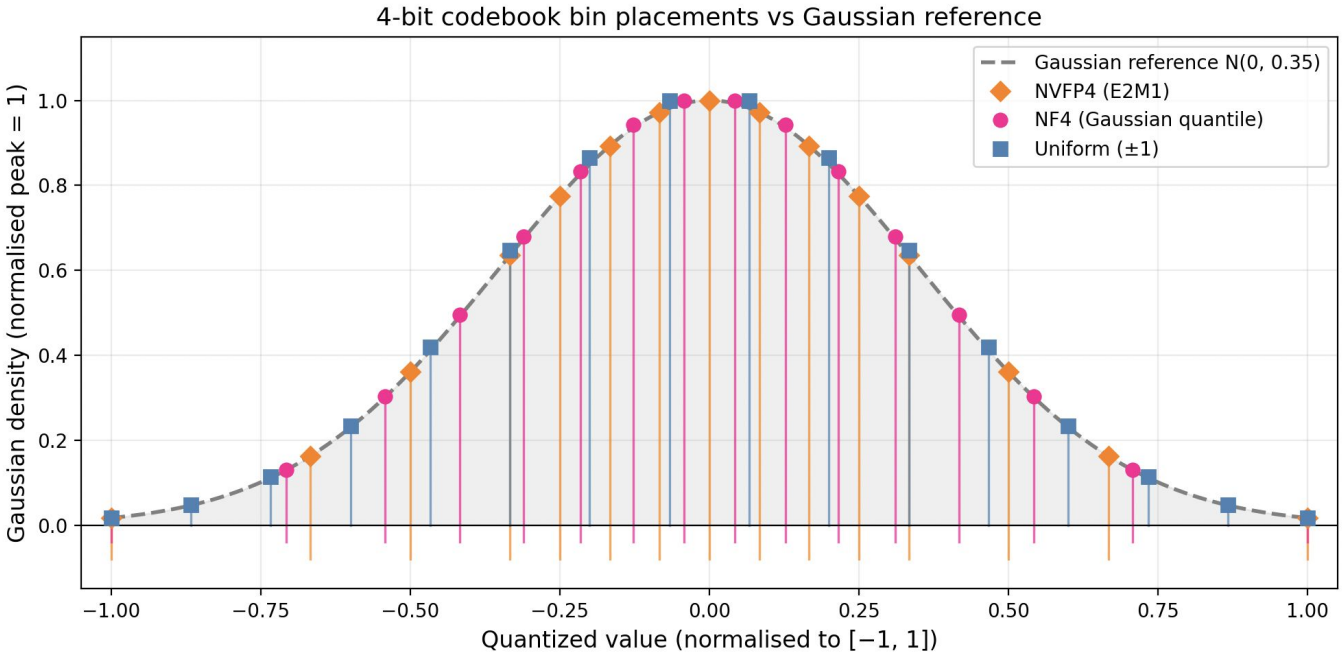
- NF4 is a 4-bit quantization format employing non-uniform quantization across 16 discrete levels.
- It better approximates the weight distribution—typically near a normal distribution—by allocating more quantization levels near zero.
- This non-uniform allocation minimizes quantization error, especially for small but critical weight values.
- NF4 is widely used in QLoRA for efficient fine-tuning of large language models.


Uniform vs NF4 Quantization Bins (4-bit)
Uniform Quantization (4-bit)
Equal bin width: 0.1333
Weight Distribution
Quantization Bins
Probability Density
Weight Value
NF4 Quantization (4-bit)
Weight Distribution
Quantization Bins
Variable bin width: denser near zero
0.0624
0.3125
Probability Density
Weight Value


Quantization Error Comparison (Uniform vs NF4)
Average Weighted Error (Uniform): 0.033334
Average Weighted Error (NF4): 0.027805
Improvement: 16.59%
Uniform Quantization Error
NF4 Quantization Error
Weight Distribution (scaled)
Absolute Quantization Error
Original Weight Value
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

FP4 (MXFP4, NVFP4)

- This method uses a codebook with 4 bit floating point numbers
- Proportional to {±0.0, ±0.5, ±1.0, ±1.5, ±2.0, ±3.0, ±4.0, ±6.0}



OUR WORK

Optimal Quantization for Matrix Multiplication

Principled methods for LLMs, VLMs, and RAG

High-rate Quantization Theory

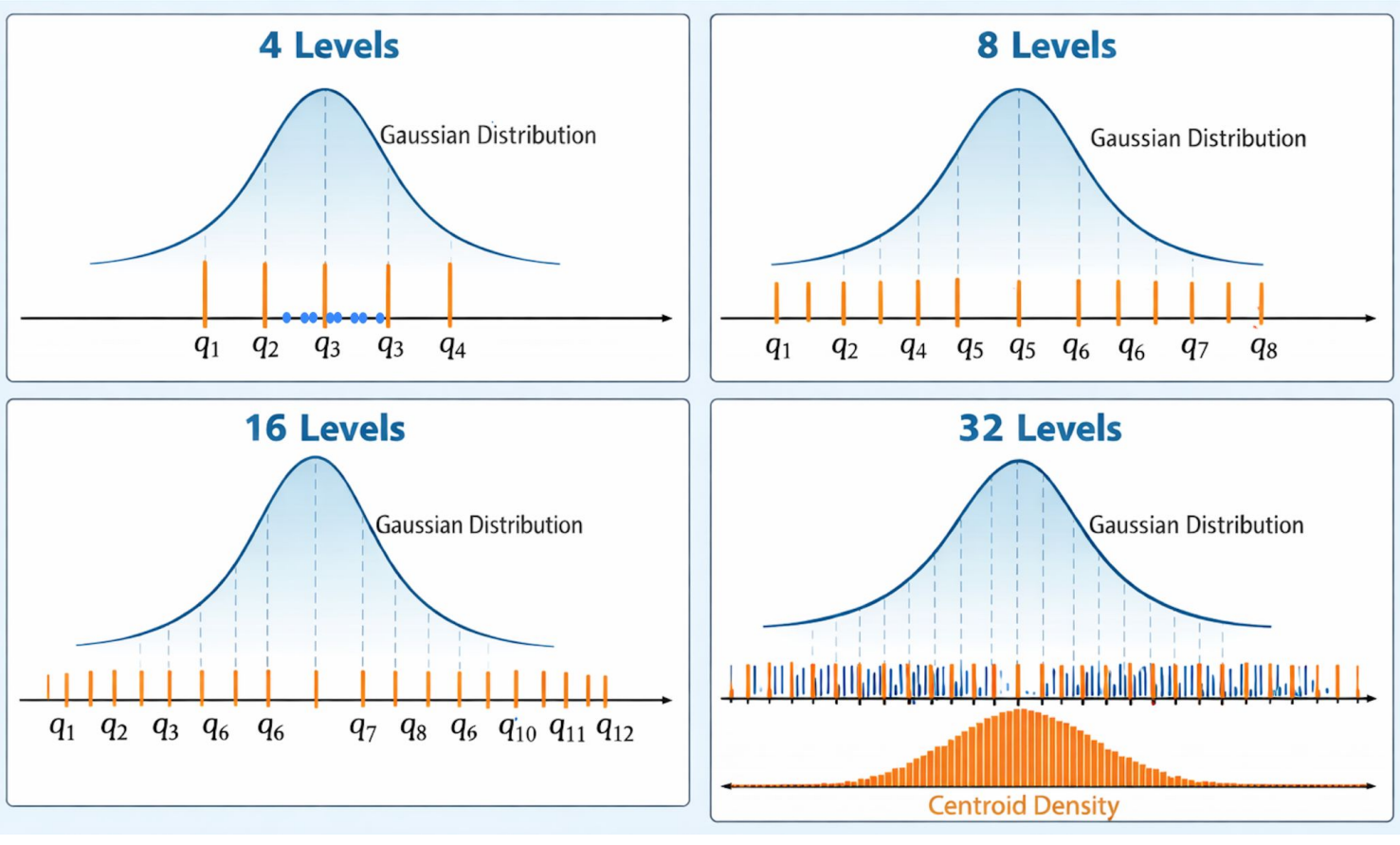
- Mean Squared Error criteria optimizes entrywise matrix approximation

$$\min_{Q(X)} \mathbb{E}[\|X - Q(X)\|_2^2], \quad Q(X) \in \{y_1, \ldots, y_M\}.$$

  Optimal quantizer that minimizes Mean Squared Error

- This can be solved **optimally** in the asymptotic regime
  - number of quantization centroids tend to infinity
  - we use a centroid distribution to describe the quantizer

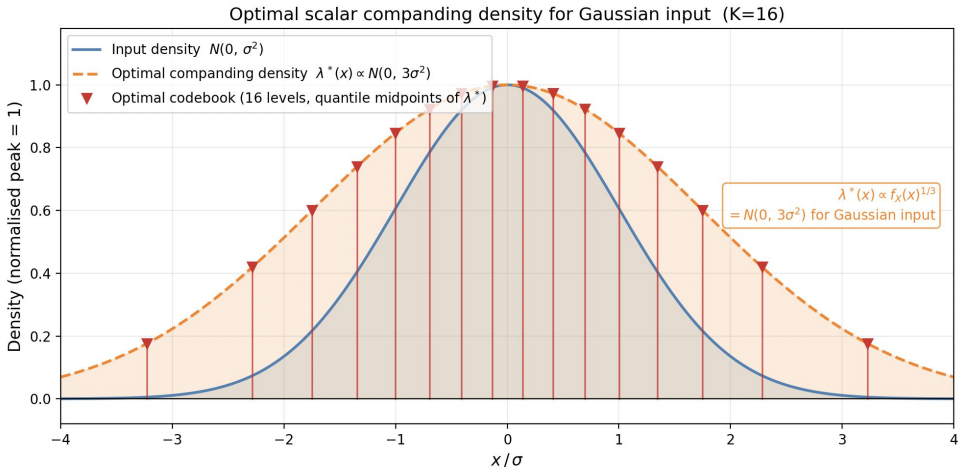
High-rate Quantization Theory



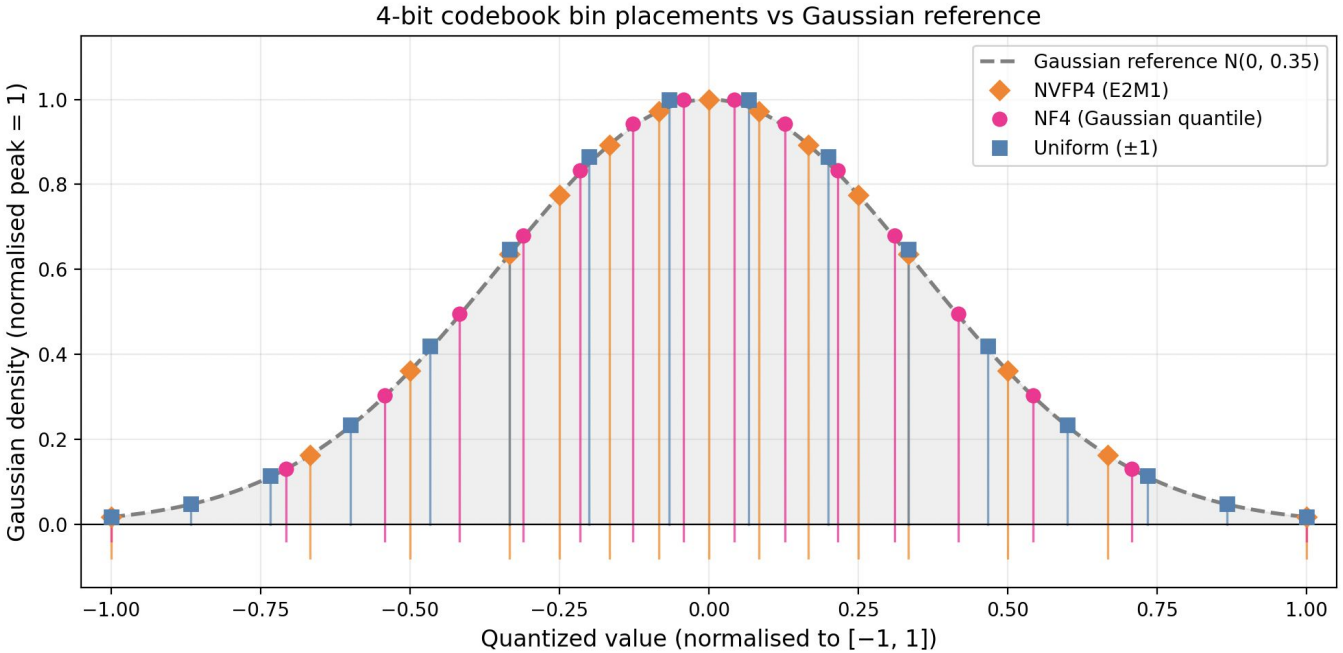
Classical Result: Quantizing a Gaussian Density

If $X \sim \mathcal{N}(0, \sigma^2)$ then, it turns out that the optimal point density is

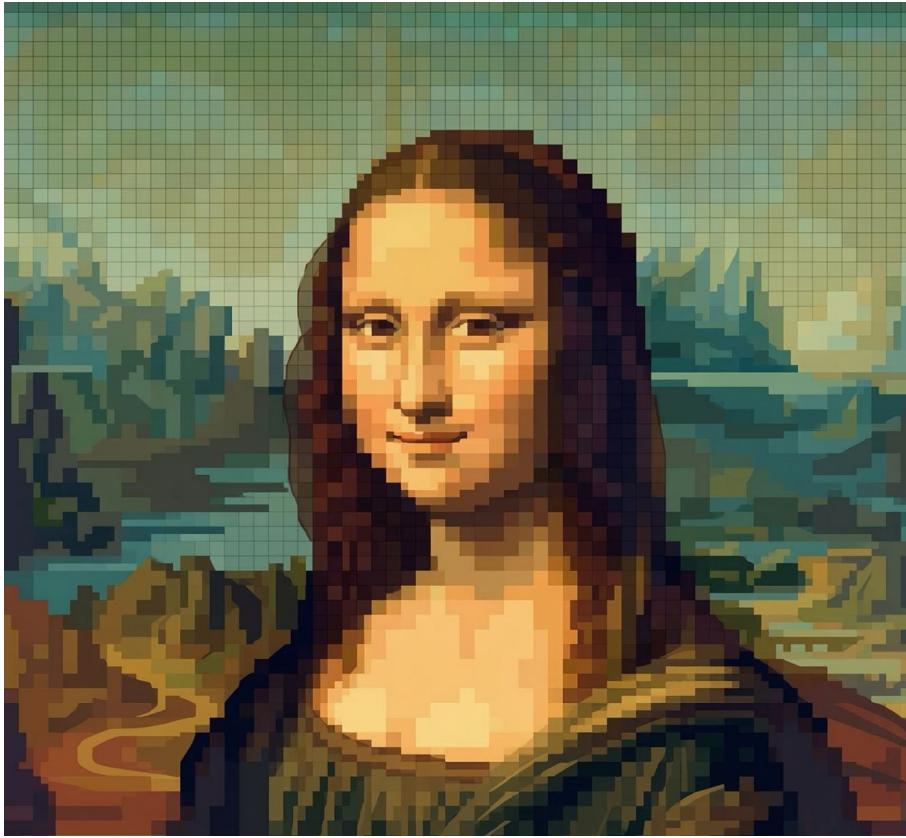
$$\lambda(x) \propto \exp\left(-\frac{x^2}{6\sigma^2}\right).$$



FP4 and NormalFloat seem to be optimal?



Entrywise approximation?



Problems with the current approach

- Mean Squared Error criteria optimizes for **entrywise distortion**

$$\min \mathbb{E}[\|X - Q(X)\|_2^2], \quad Q(X) \in \{y_1, \ldots, y_M\}.$$

- NF4, FP4 (MXFP4, NVFP4) have Gaussian-like centroid distribution so they should work well for this purpose
- However, we are more interested in the preserving the **action of linear layers**. Hence, **matrix multiplication** specific notions of distortion for quantizing LLMs/VLMs

Optimal Quantization for Matrix Multiplication

- Suppose we want to compute the product of two matrices $C = AB$
- Find the optimal quantizer that minimizes the matrix multiplication error

$$\underset{Q_X \;\; Q_Y}{\text{minimize}} \; \|C - Q_X(A)Q_Y(B)\|_F^2$$

- **Surprising Fact**:

  we can solve this problem optimally in the asymptotic regime!

Optimal Quantization for Matrix Multiplication

- Suppose we want to compute the product of two matrices $C = AB$
- Find the optimal quantizer that minimizes the matrix multiplication error

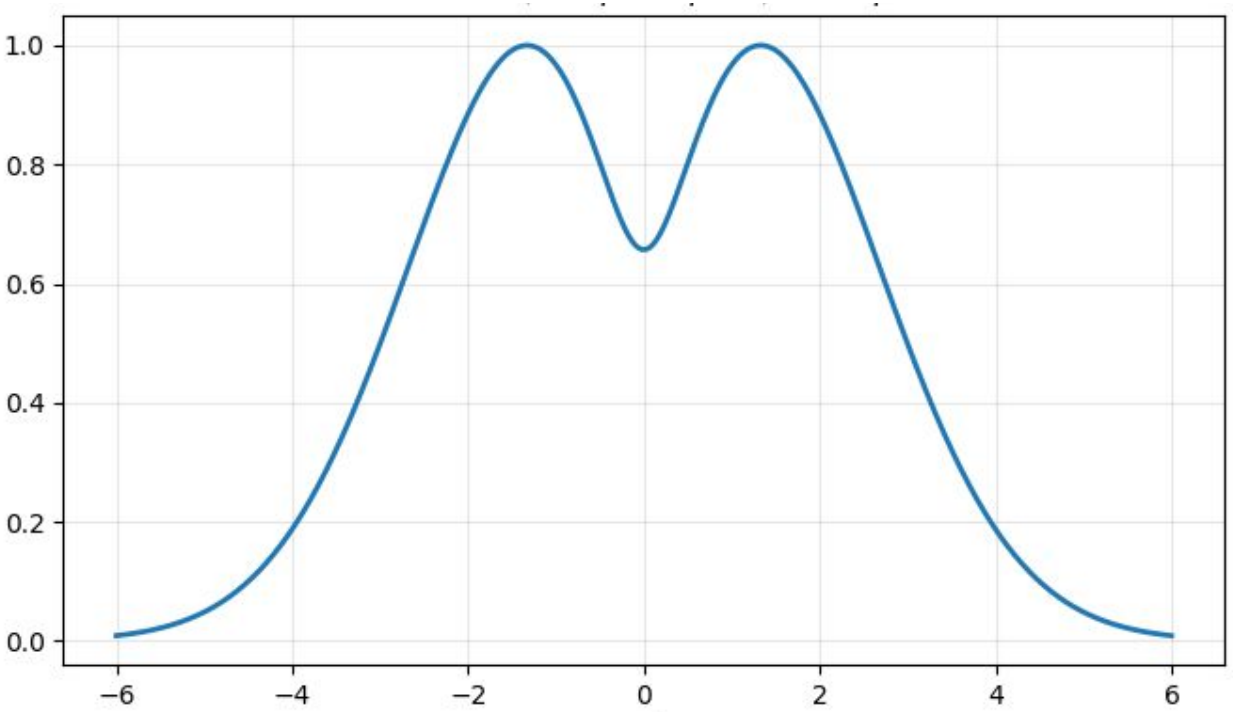
$$\underset{Q_X \;\; Q_Y}{\text{minimize}} \; \|C - Q_X(A)Q_Y(B)\|_F^2$$

- **Solution:**

  Depends on the correlation between the entries of A and B

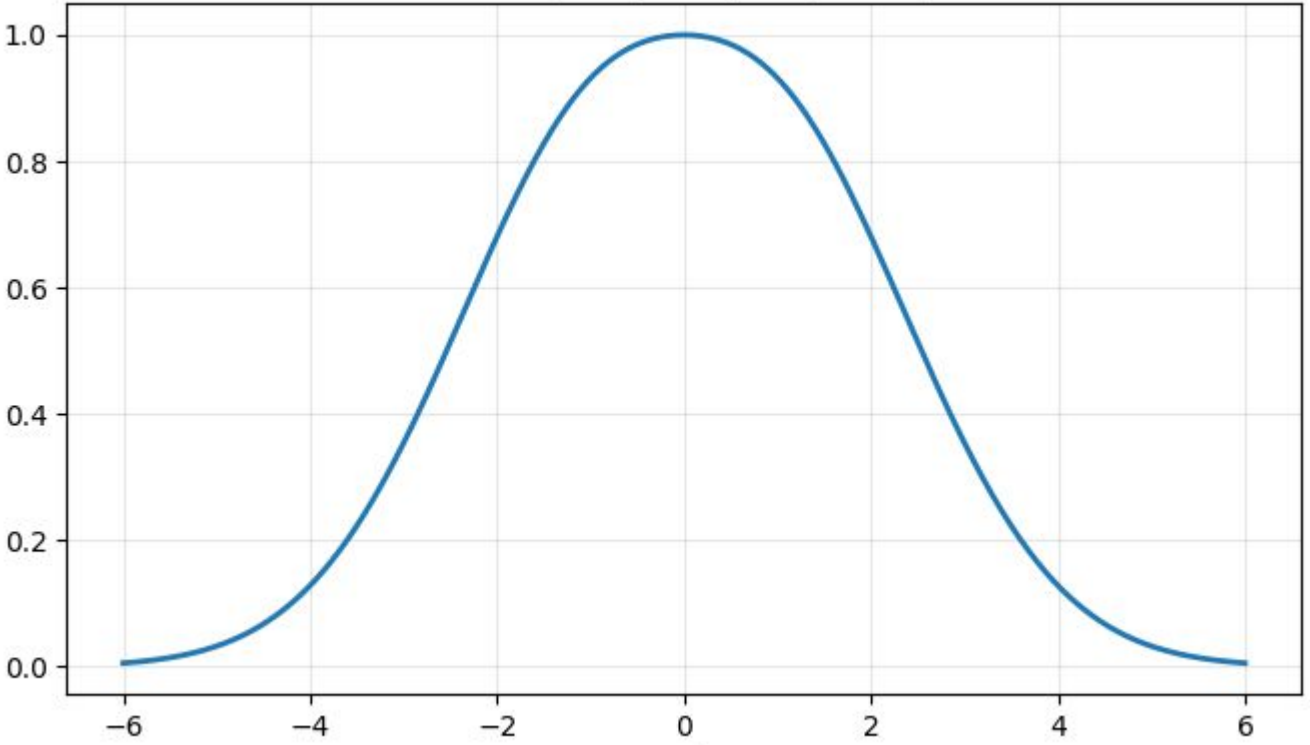
Optimal Quantization for Matrix Multiplication

- Optimal centroid density: **high correlation**



Optimal Quantization for Matrix Multiplication

- Optimal centroid density: **low correlation**



The Gaussian Case

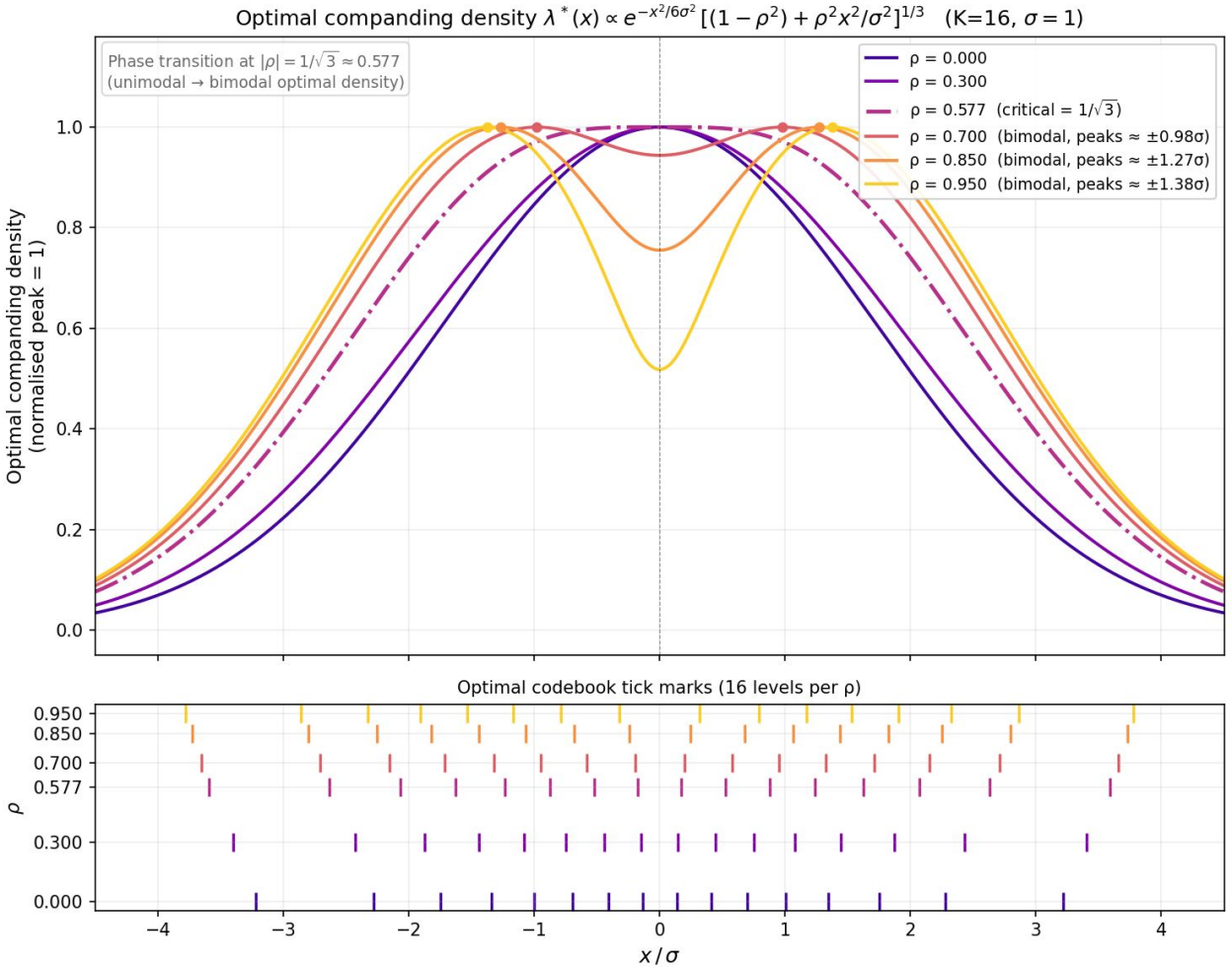
If we assume

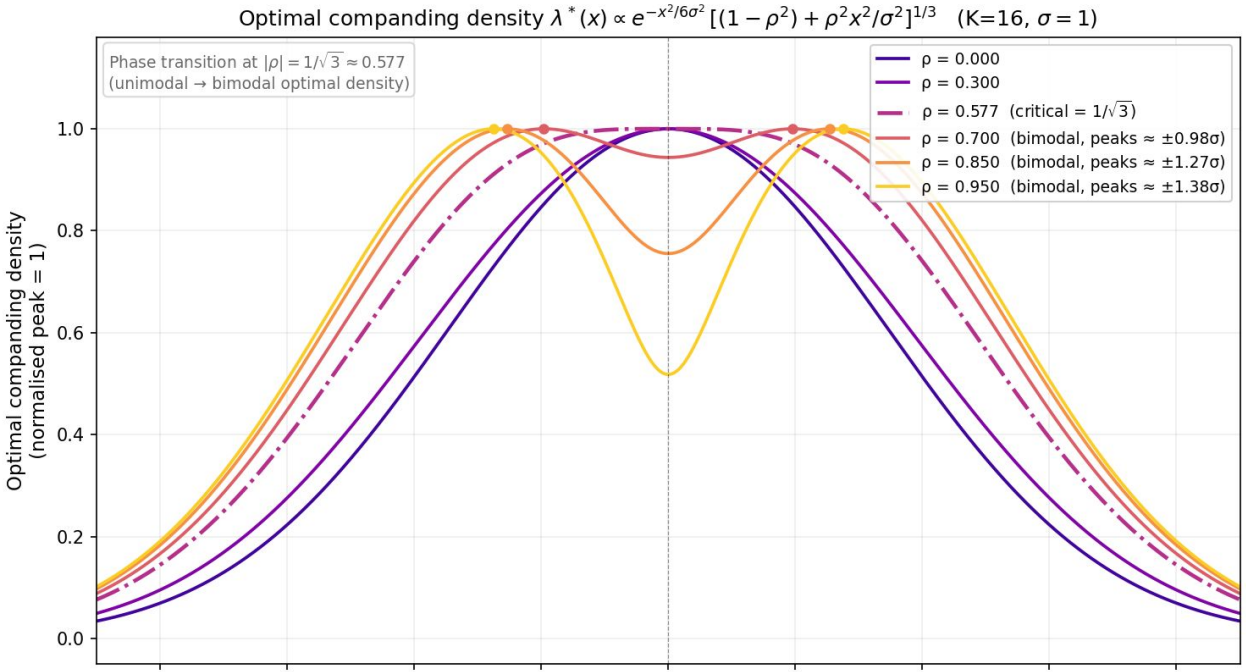
$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{bmatrix} \right)$$

then the optimal densities become

$$\lambda_X(x) := \exp\left(-\frac{x^2}{6\sigma_X^2}\right)\left((1-\rho^2)+\rho^2\frac{x^2}{\sigma_X^2}\right)^{1/3}$$

$$\lambda_Y(y) := \exp\left(-\frac{y^2}{6\sigma_Y^2}\right)\left((1-\rho^2)+\rho^2\frac{y^2}{\sigma_Y^2}\right)^{1/3}.$$

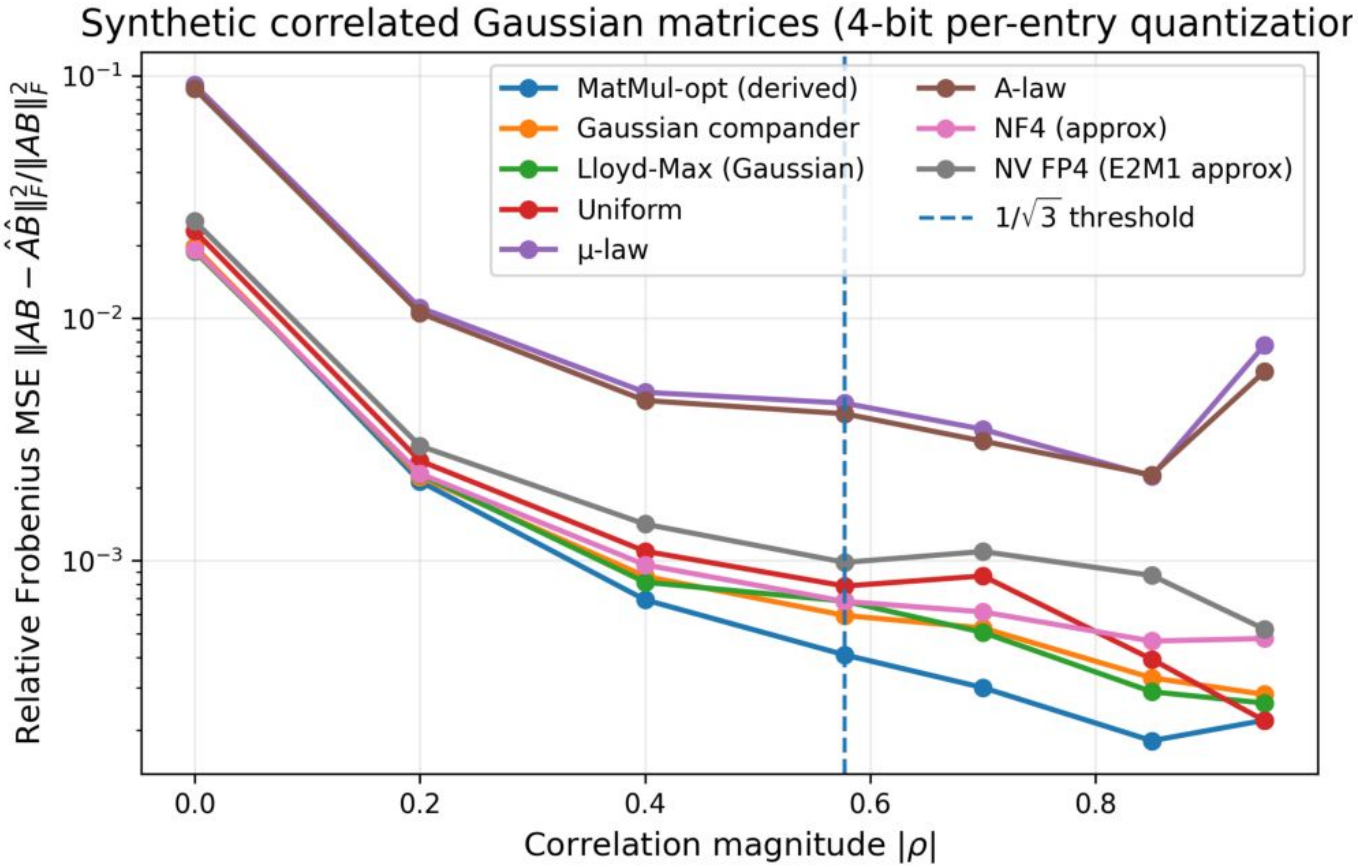

Optimal companding density $\lambda^*(x) \propto e^{-x^2/6\sigma^2}\,[(1-\rho^2)+\rho^2 x^2/\sigma^2]^{1/3}$ (K=16, $\sigma=1$)
Phase transition at $|\rho| = 1/\sqrt{3} \approx 0.577$
(unimodal → bimodal optimal density)
ρ = 0.000
ρ = 0.300
ρ = 0.577 (critical = $1/\sqrt{3}$)
ρ = 0.700 (bimodal, peaks ≈ ±0.98σ)
ρ = 0.850 (bimodal, peaks ≈ ±1.27σ)
ρ = 0.950 (bimodal, peaks ≈ ±1.38σ)
Optimal companding density
(normalised peak = 1)
0.0
0.2
0.4
0.6
0.8
1.0
Optimal codebook tick marks (16 levels per ρ)
ρ
0.950
0.850
0.700
0.577
0.300
0.000
−4
−3
−2
−1
0
1
2
3
4
x/σ



- **Intuition**: when correlation is large, **small** entries are frequently multiplied with **small** entries and become smaller, hence can be ignored

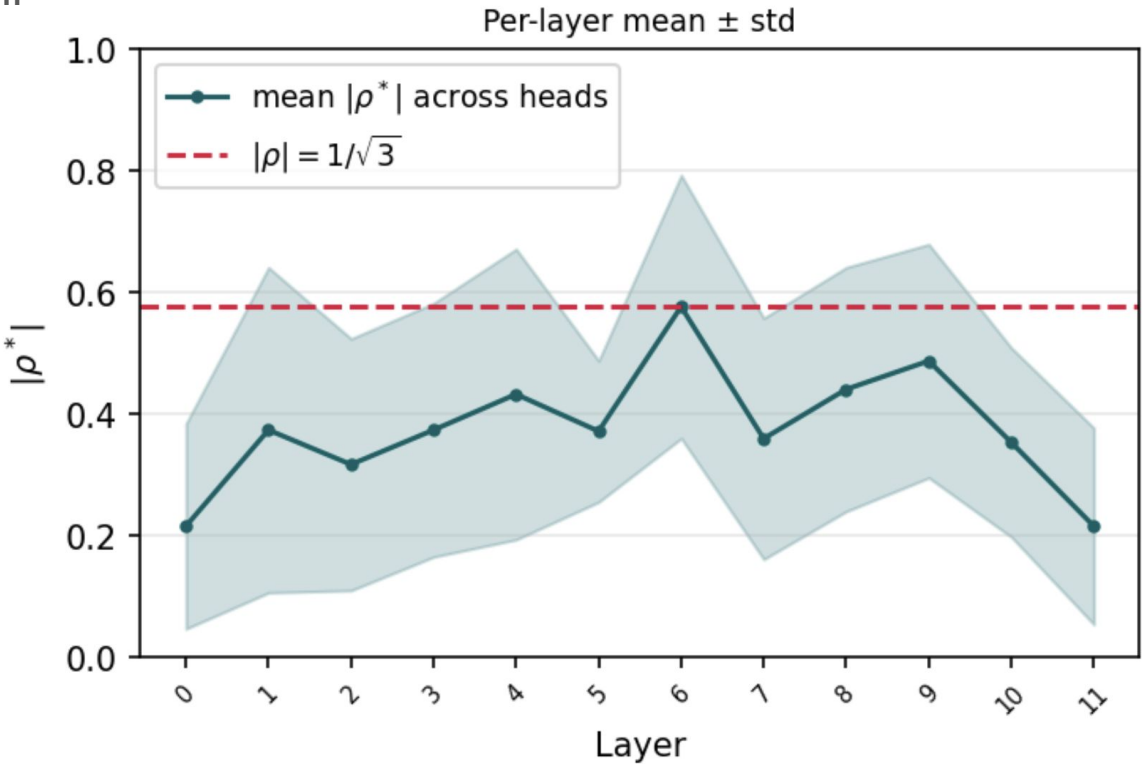
Implications

- NF4/NVFP4 based quantization methods are analogous to assuming that the entries in matrices being quantized are completely uncorrelated
- When quantizing VLMs this is a dubious assumption
  - E.g. key and query activations in self-attention mechanism are likely to be correlated
- By tuning rho and utilizing the derived optimal matrix multiplication quantizer we can more accurately quantize weights and activations in VLMs


Synthetic correlated Gaussian matrices (4-bit per-entry quantizatior
MatMul-opt (derived)
Gaussian compander
Lloyd-Max (Gaussian)
Uniform
μ-law
A-law
NF4 (approx)
NV FP4 (E2M1 approx)
$1/\sqrt{3}$ threshold
Relative Frobenius MSE $\|AB - \hat{A}\hat{B}\|_F^2/\|AB\|_F^2$
10^{-1}
10^{-2}
10^{-3}
0.0
0.2
0.4
0.6
0.8
Correlation magnitude $|\rho|$

Numerical Correlations of Matrix Multiplies in LLMs

- GPT2 small

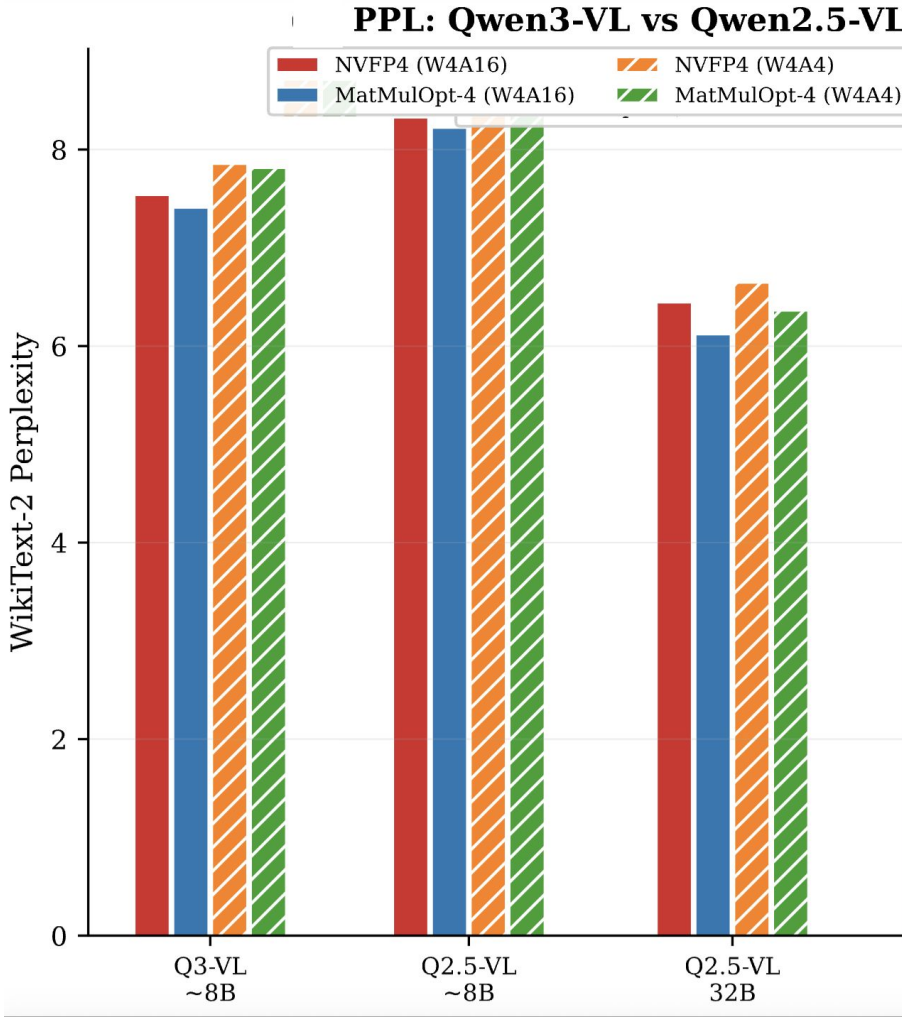


Win Rate in LLM Matrix Multiplies

Model	ρ-tuned vs FP8 (% wins)
GPT-2 Small (12 layers, 12 heads)	100%
GPT-2 Medium (24 layers, 16 heads)	100%
GPT-2 Large (36 layers, 20 heads)	100%
GPT-2 XL (48 layers, 25 heads)	100%
Qwen3-0.6B (28 layers, 8 heads)	98.7%
Qwen3-1.7B (28 layers, 8 heads)	98.7%
Qwen3-8B (36 layers, 8 heads)	86.1%

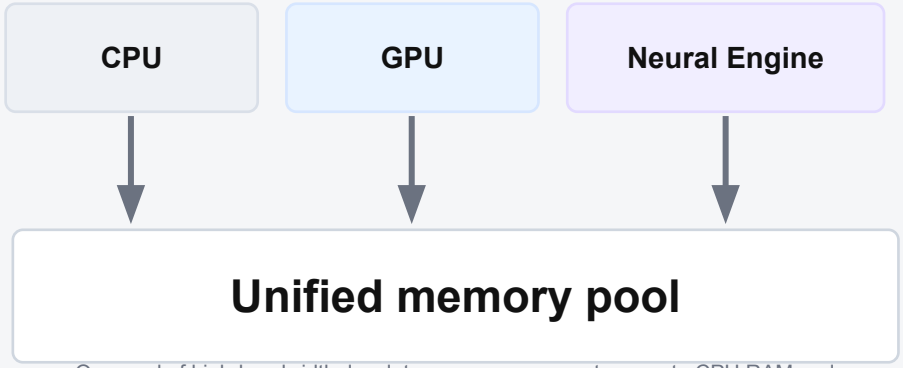
Perplexity Benchmark

- 8B and 32B Qwen models
- two variants:
  - 4 bit weight 16 bit activation
  - 4 bit weight 4 bit activation



Apple Inference: Unified Memory

CPU, GPU, and Neural Engine can access one shared memory pool



One pool of high-bandwidth, low-latency memory — not separate CPU RAM and GPU VRAM.

Mac Studio (M3 Ultra config)

On-desk inference node with unusual memory capacity

512GB
max unified memory

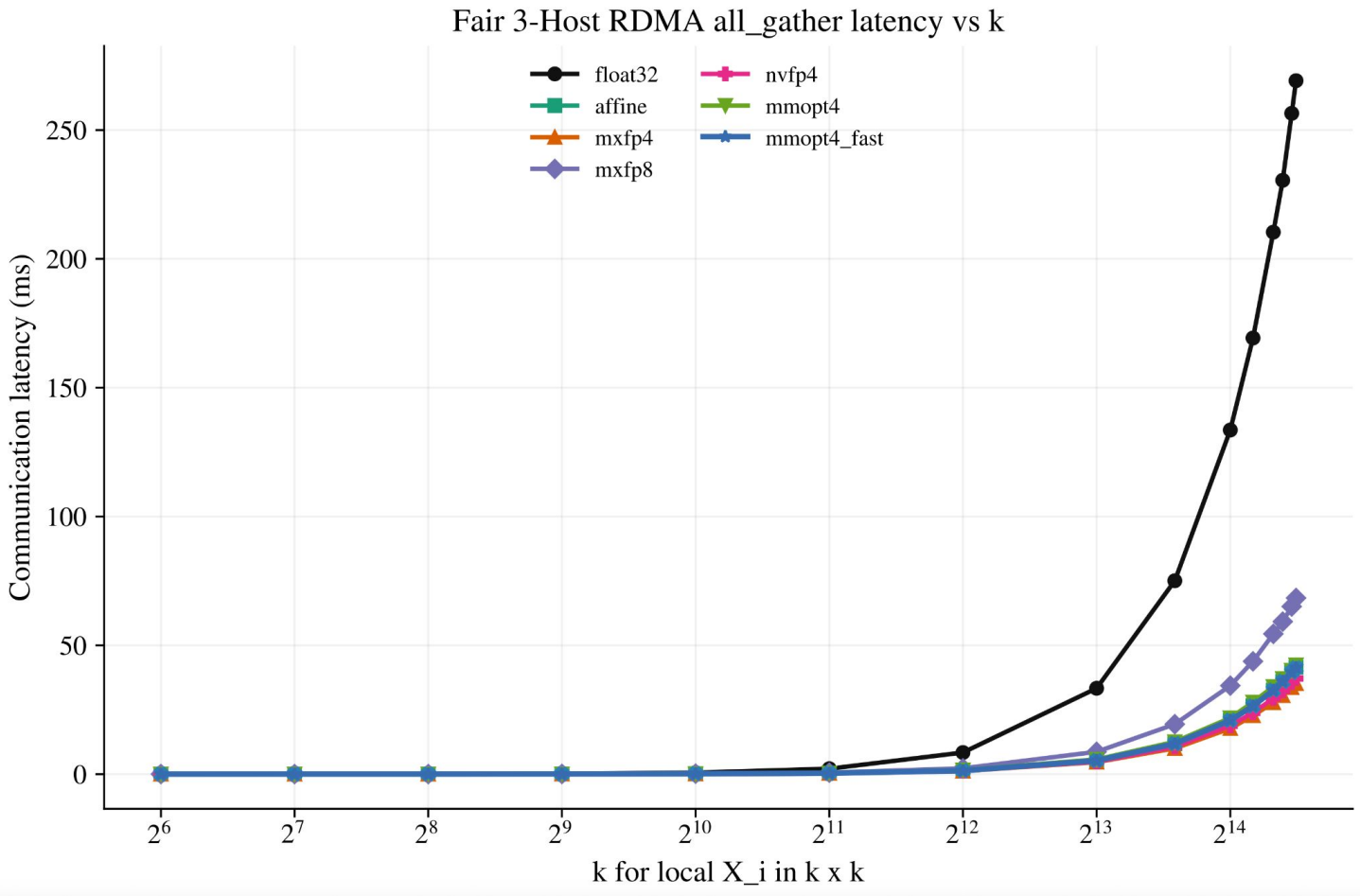
819GB/s
M3 Ultra memory bandwidth

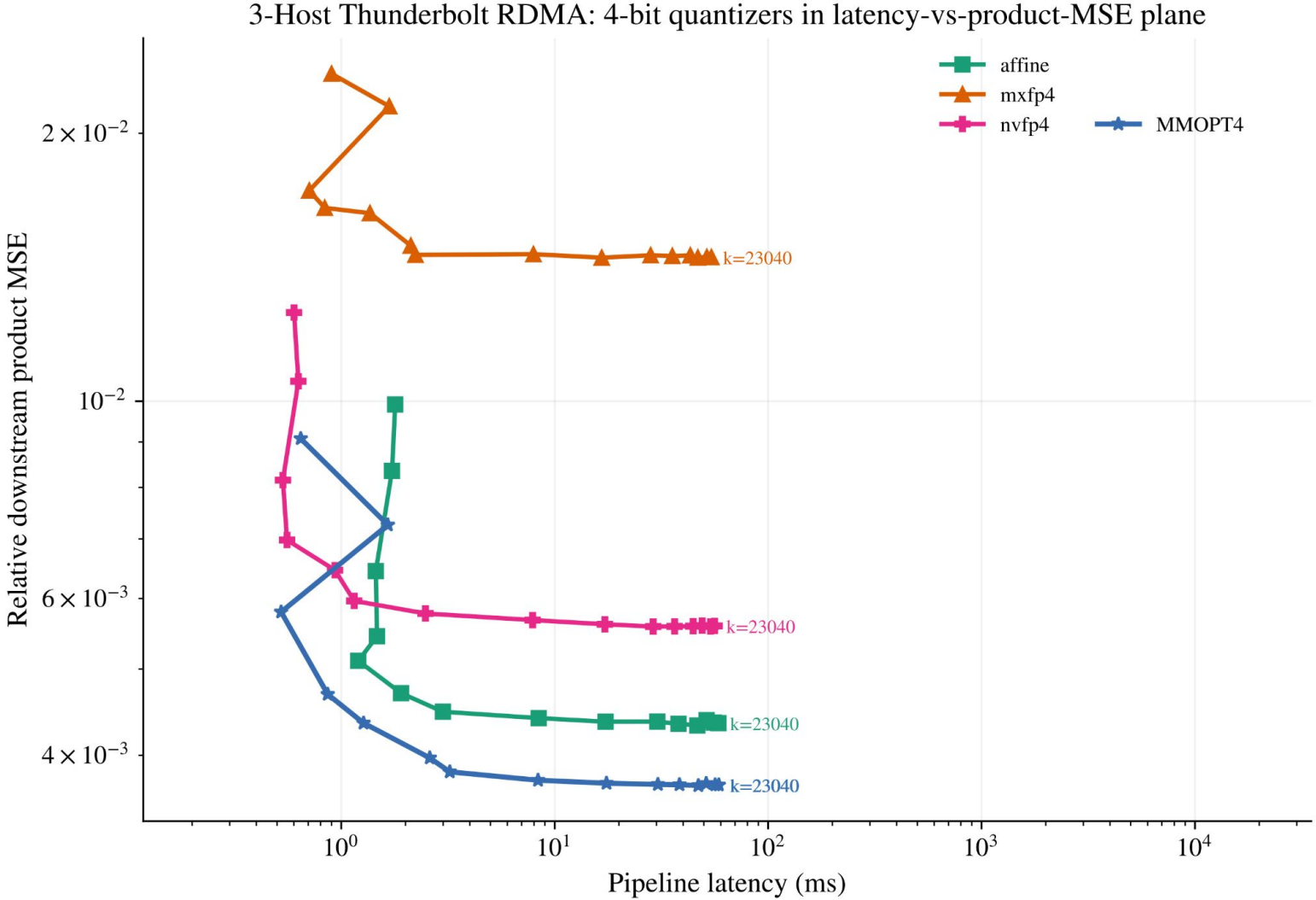
MLX
optimized for unified memory

compare with Nvidia H200: 141 GB memory
4.8 TB/s memory bandwidth

Apple 2025 Mac Studio / M3 Ultra; Apple Open Source MLX

Latency Benchmarks: Matrix Multiplication Optimal (MMOPT4) vs all



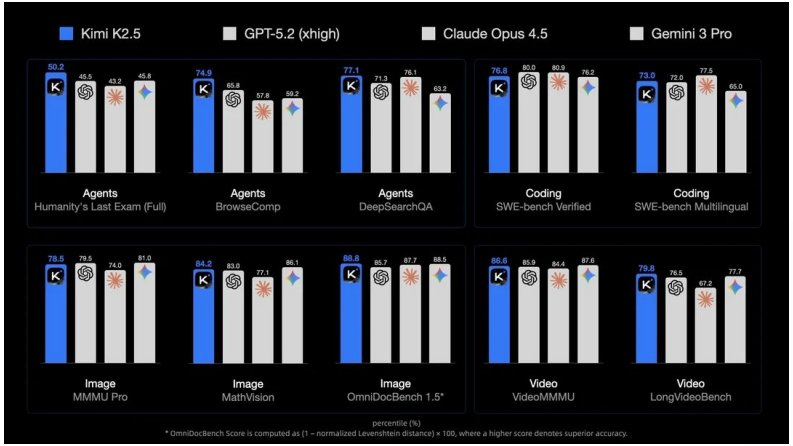

3-Host Thunderbolt RDMA: 4-bit quantizers in latency-vs-product-MSE plane
affine
mxfp4
nvfp4
MMOPT4
k=23040
k=23040
k=23040
k=23040
Relative downstream product MSE
2×10^{-2}
10^{-2}
6×10^{-3}
4×10^{-3}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
Pipeline latency (ms)

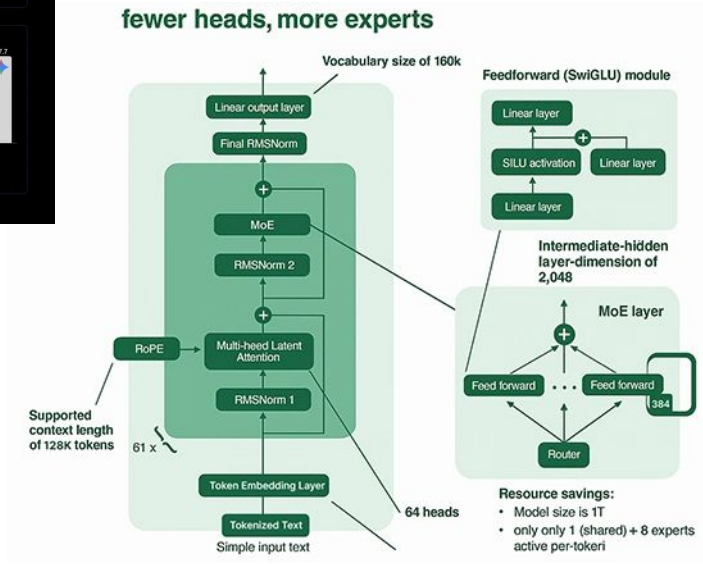
Distributed Inference

- **exo** open-source distributed inference library

Kimi K2.5 Open-Source LLM: 1.1 Trillion Parameters

Architecture	Mixture-of-Experts (MoE)
Total Parameters	1T
Activated Parameters	32B
Number of Layers (Dense layer included)	61
Number of Dense Layers	1
Attention Hidden Dimension	7168
MoE Hidden Dimension (per Expert)	2048
Number of Attention Heads	64
Number of Experts	384
Selected Experts per Token	8
Number of Shared Experts	1
Vocabulary Size	160K
Context Length	256K
Attention Mechanism	MLA
Activation Function	SwiGLU
Vision Encoder	MoonViT
Parameters of Vision Encoder	400M





Moonshot AI. January 206

Kimi K2.5 Inference

Mac Studio Cluster
24 token/s
64K context size
total energy consumption ~ 200W

Comparison
8 x Nvidia H100 server
~ 75 token/s
total energy consumption ~ 7500W

1x Nvidia 6000 Pro 96GB GPU + offload to CPU
12 t/s token generation
total energy consumption ~ 1500W

Up Next: Advanced Quantization Techniques for LLMs

Advanced Quantization Methods for LLMs

- activation aware, AWQ (Lin et al., 2024)

  $W \approx \hat{W}$ is not enough; we need $XW \approx X\hat{W}$ for calibration data X
- low-rank, LoftQ, Yi et al., 2023, Saha et al., 2023,
- transform domain, Quip - Chee et al., 2024
- dictionary based, GPTQ - Frantar et al., 2023
- quantization lattices, Quip# - Tseng et al., 2024
- use the curvature of the loss, OPTQ - Frantar et al., 2023
- **mixed-precision low-rank**, Saha et al., 2023, 2024

Outlier Separation: LLM.int8 (Dettmers et al., 2022)

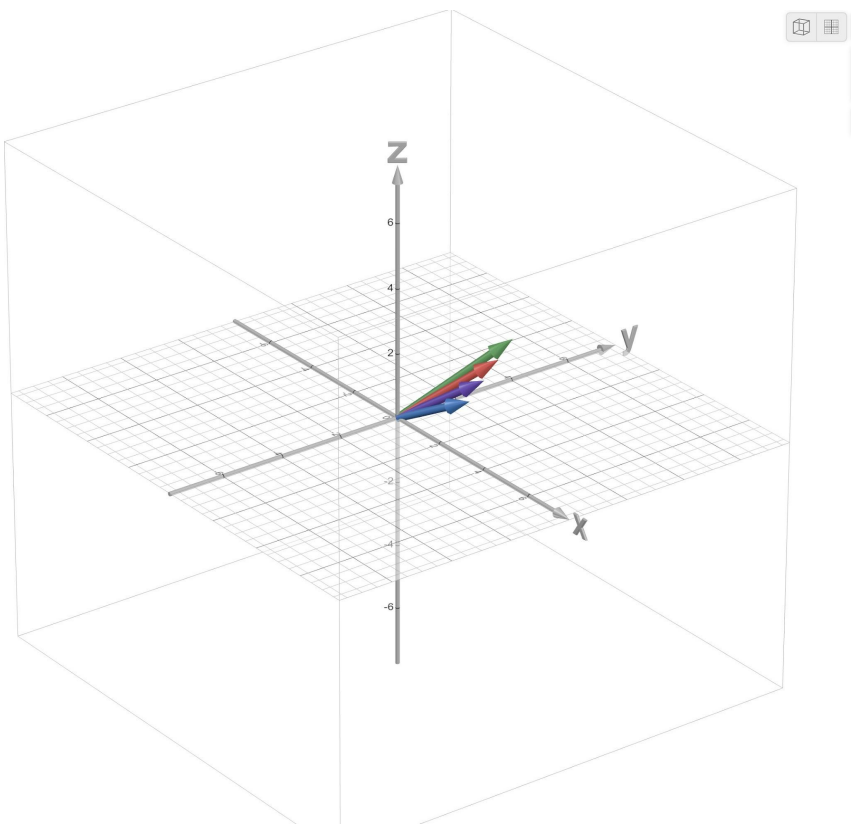
LLM.int8 quantizes weights and activations to int8 for efficiency.

However, activations and weights follow an approximate Gaussian distribution.
6-sigma threshold: Nearly all (99.7%) activations/weights lie within $\pm 6\sigma$.

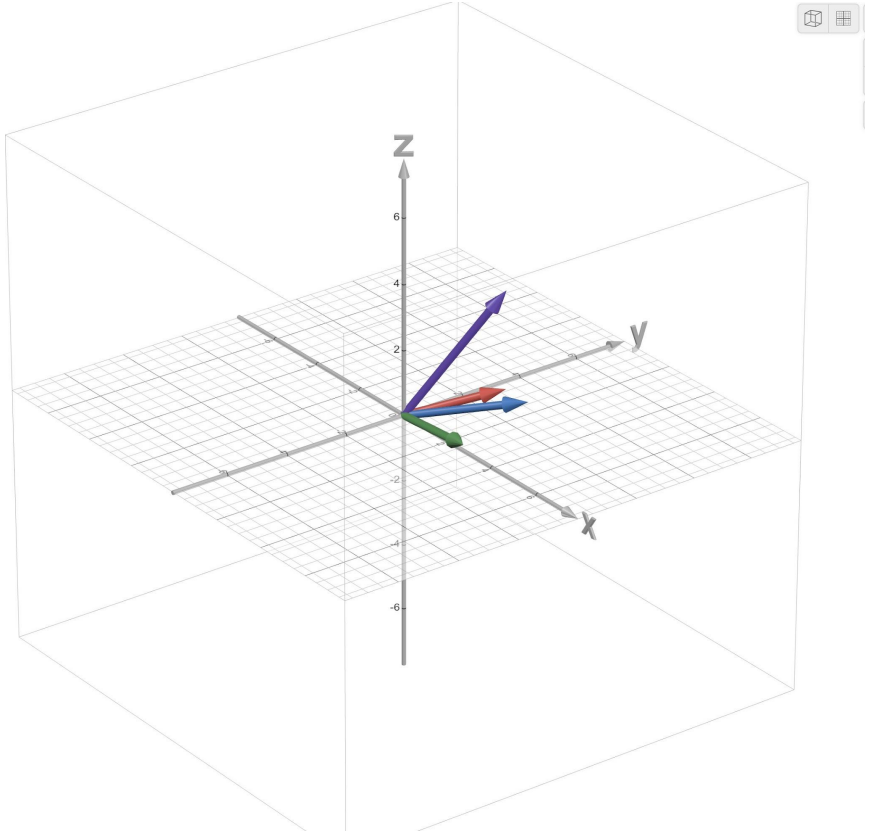
Values beyond 6σ are deemed **outliers** and computed in higher precision (FP16).

This mixed precision approach preserves overall model accuracy while reducing memory usage.

Rank of a matrix: Dimension of the vector space generated by its columns

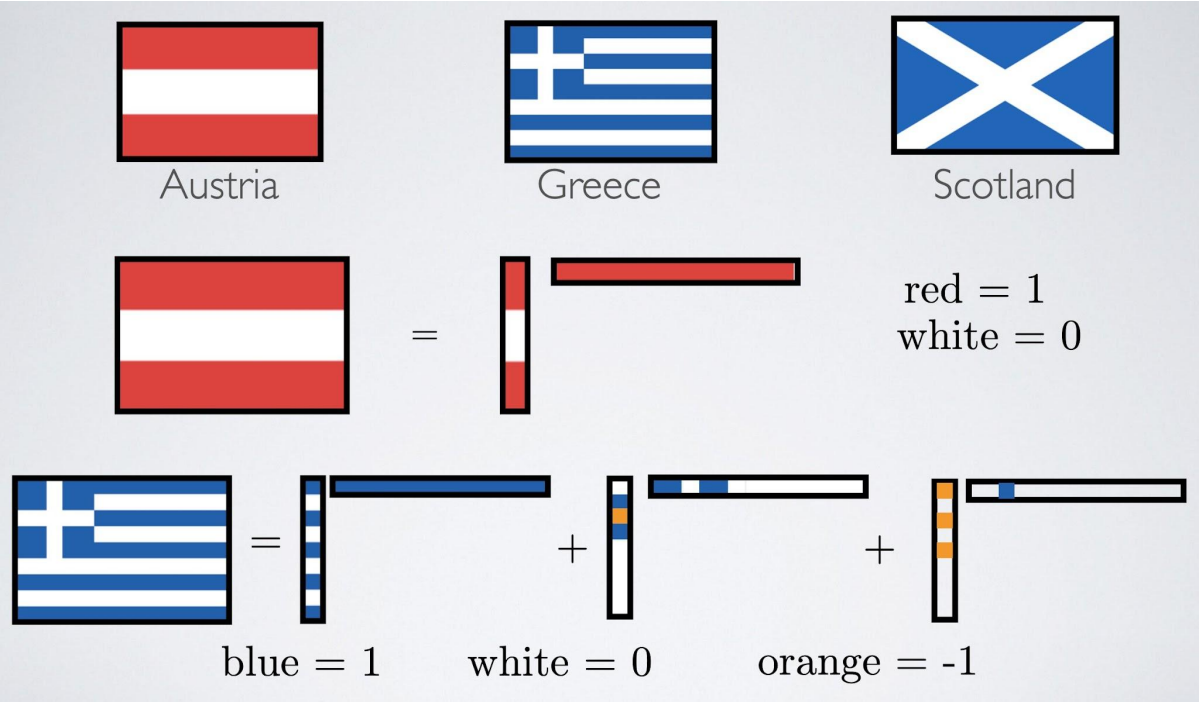


rank equals 2



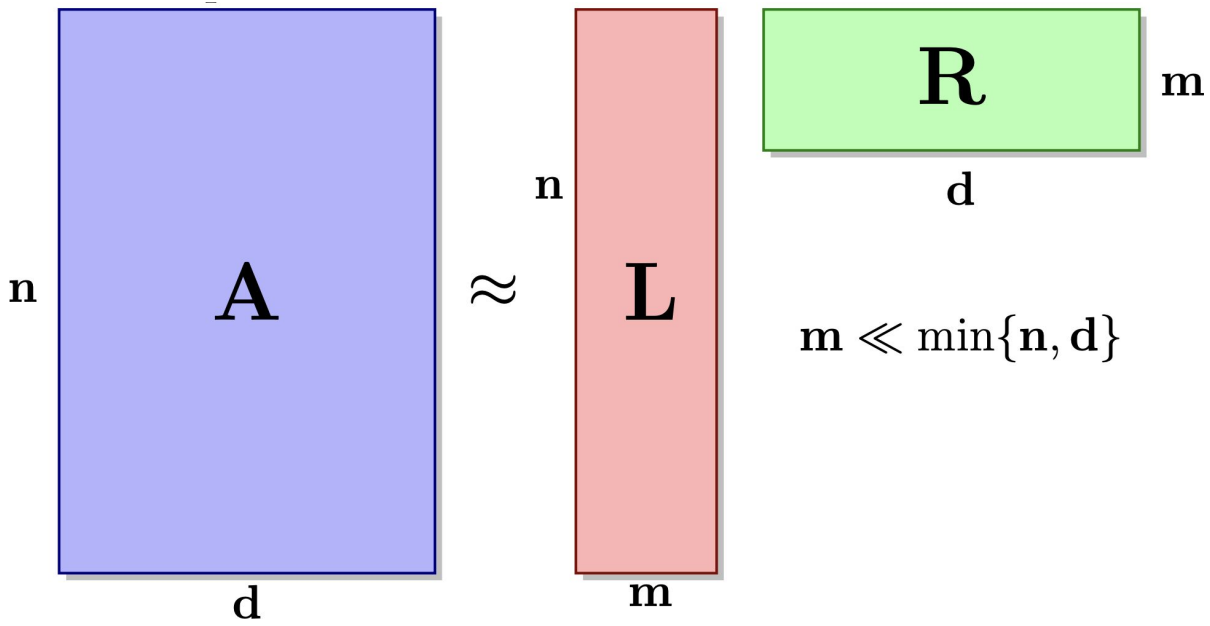
rank equals 3

What do low-rank matrices look like?



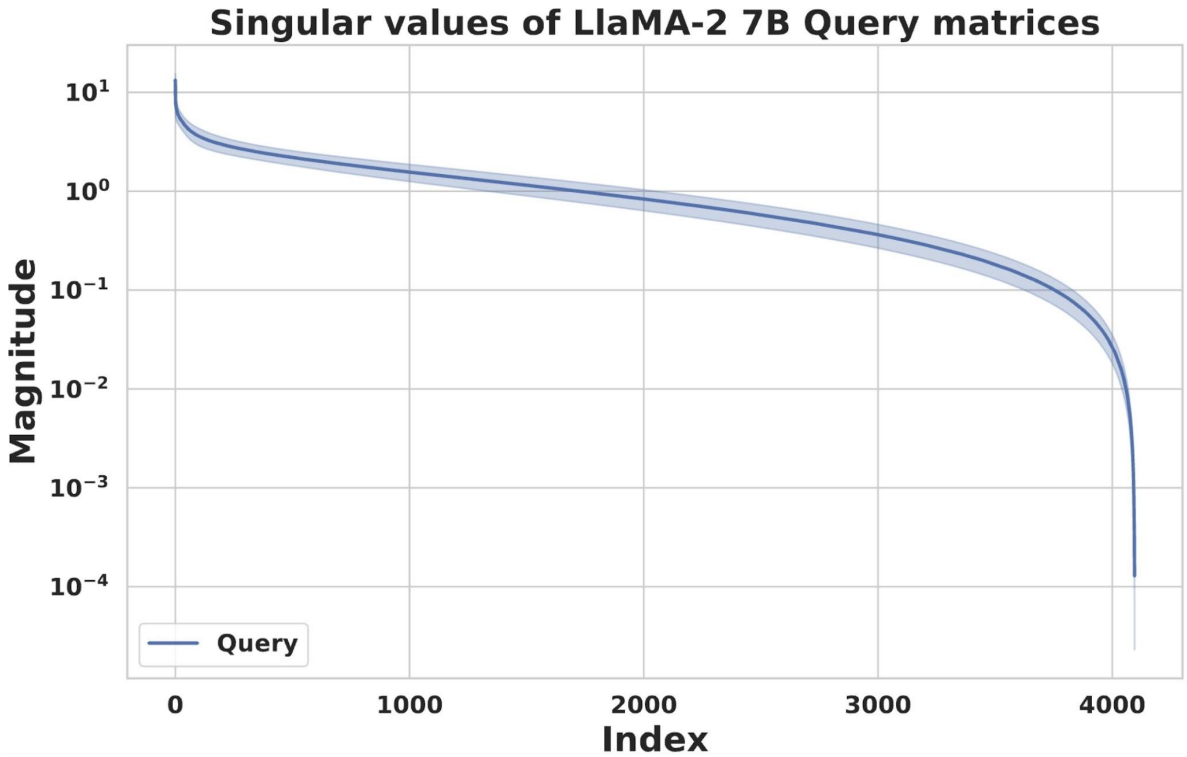
- extremely low-rank matrices are aligned to the grid (source: A. Townsend)

Low-rank approximation



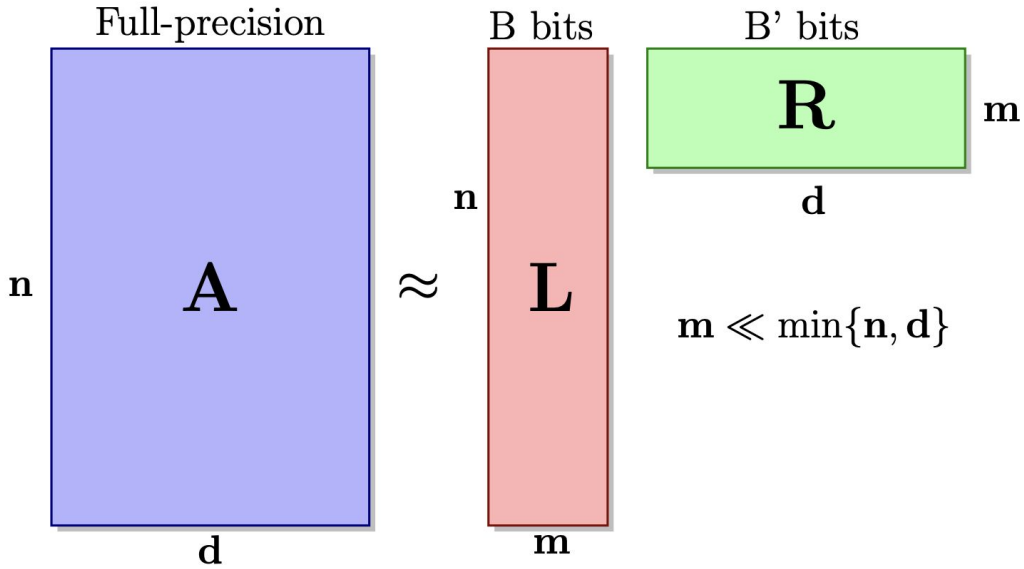
low-rank approximation is described using much fewer numbers

LLM weights are approximately low-rank



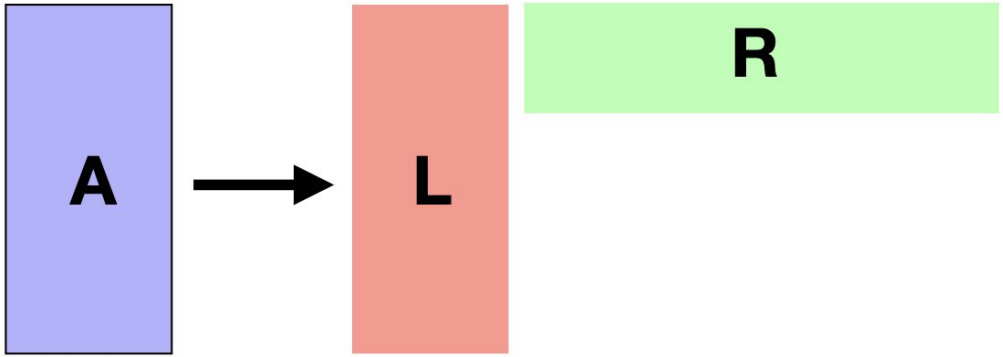
singular value decomposition reveals near low-rank matrices

Low-Precision Low-Rank (LPLR) Decomposition



- our algorithm finds this decomposition with near-optimal error
  Saha, Srivastava, Pilanci, Matrix Compression via Randomized Low Rank and Low Precision Factorization, NeurIPS 2023

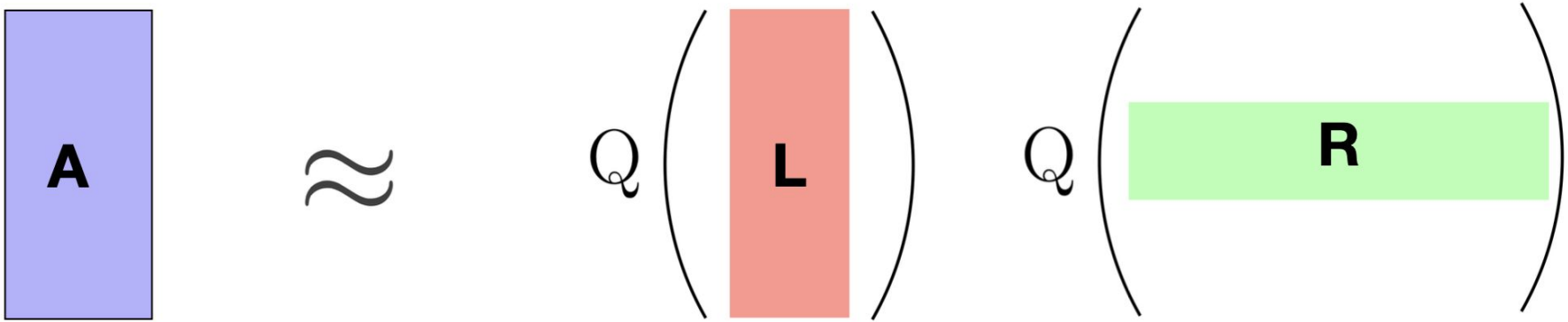
  2026 update: Matrix Multiplication Optimal quantizer (MMOPT) for L and R



original weight

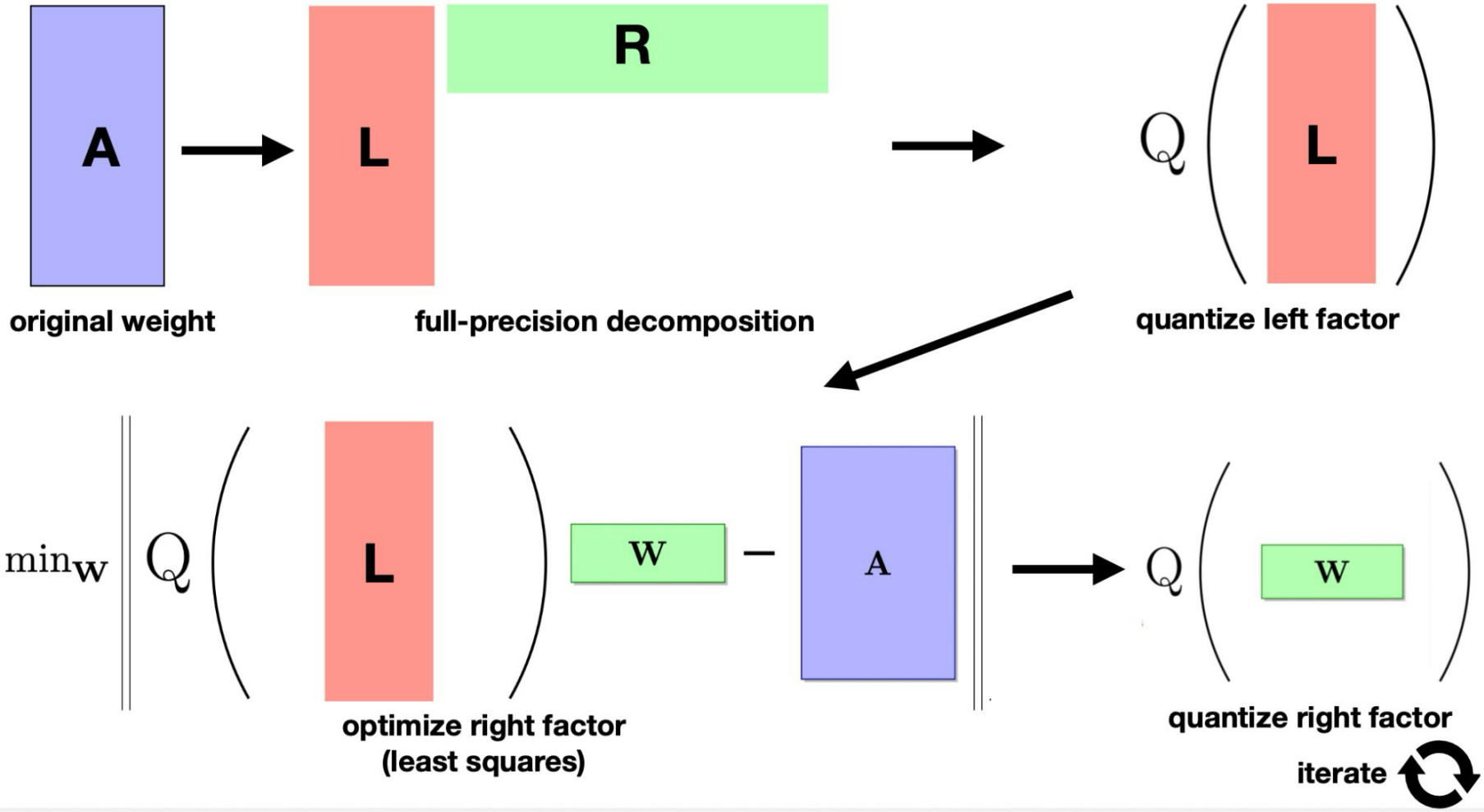
full-precision decomposition

not optimal : large errors!



quantize left factor

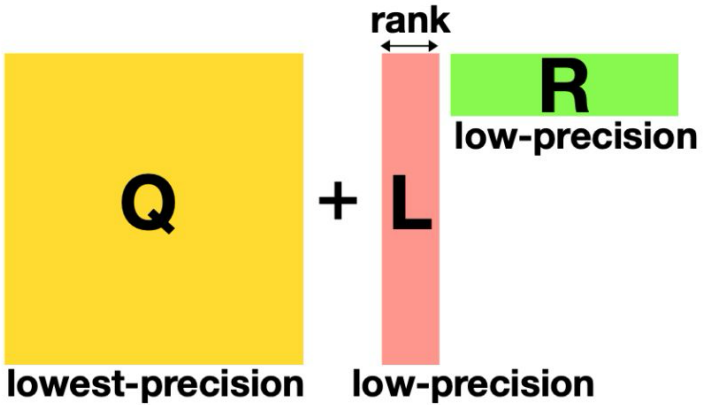
quantize left factor


A
L
R
Q
L
original weight
full-precision decomposition
quantize left factor
$\min_{\mathbf{W}} \left\| Q\left(\right) \right.$
L
W
A
optimize right factor
(least squares)
Q
W
quantize right factor
iterate

General Low-Precision Decomposition

Given a weight matrix A, decompose as

$$A \approx Q + LR$$

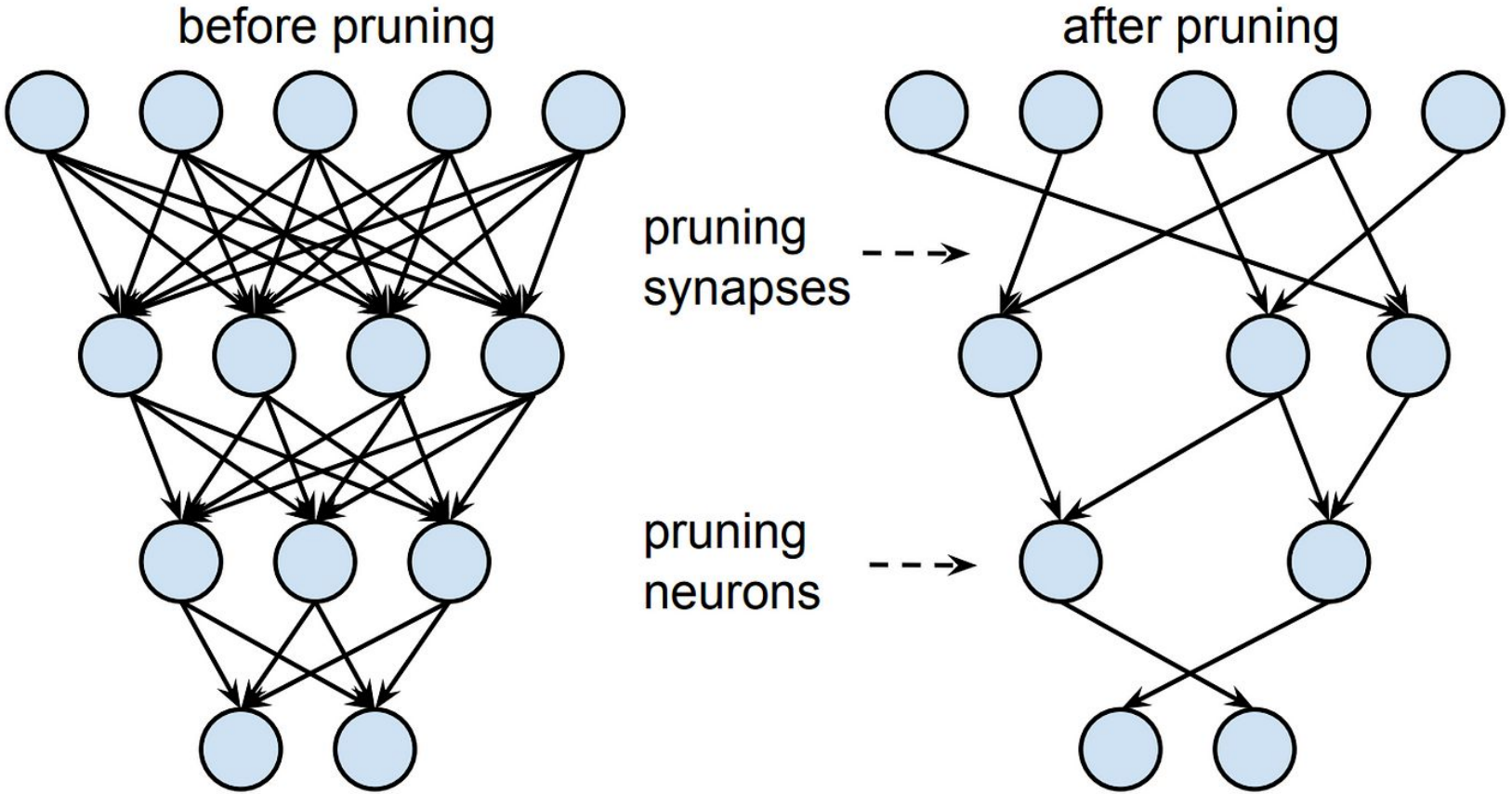


- Q is a full-size low-precision matrix, e.g., 2-bit, NormalFloat, Outlier Separation etc.
- L is a tall moderate-precision matrix, e.g., 8-bit
- R is a wide moderate-precision matrix, e.g., 4-bit
- we (approximately) solve the optimization problem

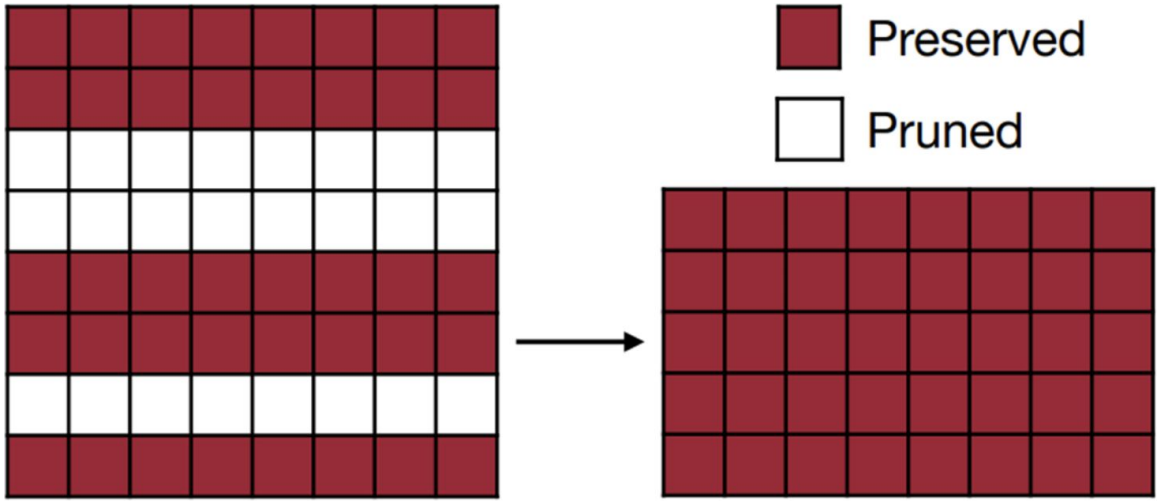
$$\min_{Q,L,R \in \mathbf{low-precision}} \|A - Q - LR\|$$

Up Next: Pruning for LLMs

Pruning


before pruning
after pruning
pruning
synapses
pruning
neurons

Pruning



Pruning: removing parameters
Individual parameters, or parameters in groups such as by neurons or channels Typical strategy:

1. Evaluate the importance of each unit (neuron/channel).
2. Rank the units according to their importance
3. Remove the least important neuron(s)

Overview of Common Neural Network Pruning Methods

1. **Magnitude-based Pruning**

   Remove smallest magnitude weights:

   $$|w_{ij}| < \text{threshold}$$

   Pros: Simple, efficient.

   Cons: May miss inference-time importance.

2. **Gradient-based Pruning**

   Use gradients to prune during training:

   $$|w_{ij} \cdot g_{ij}| < \text{threshold}$$

   Pros: Aligns pruning with training objectives.

   Cons: Requires gradients, expensive at scale.

3. Optimal Brain Surgeon (LeCun et al. 1989; Hassibi and Stork 1992)

Use Hessian for weight importance:

$$\Delta w = -H^{-1} \frac{\partial L}{\partial w}$$

Pros: High accuracy.

Cons: Computationally intensive.

4. Wanda (Weights and Activations) (Sun et al., 2023)

Prune weights with smallest product of weights and activations:

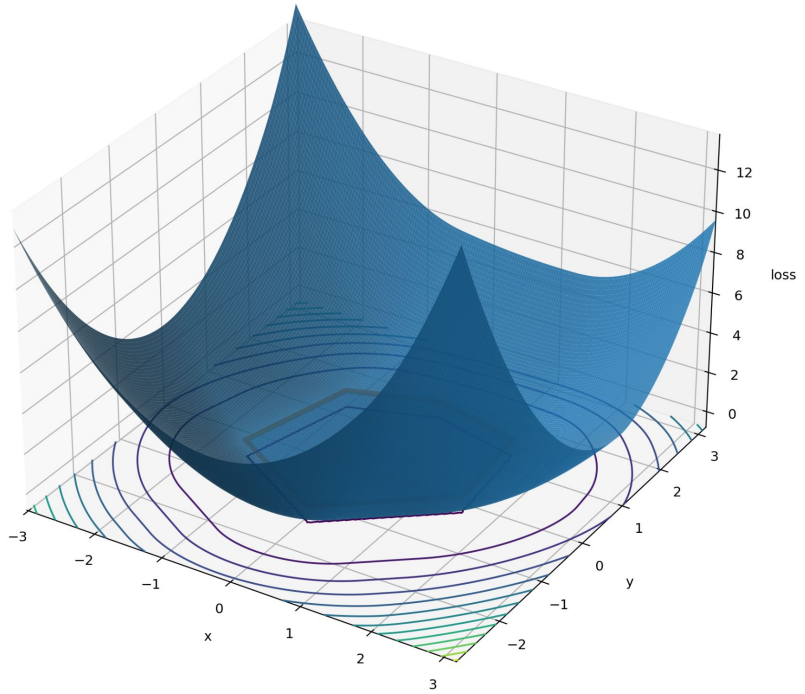
$$|w_{ij} \cdot a_i| < \text{threshold}$$

Pros: No retraining, efficient for large-scale LLMs.

Cons: Depends on activation representativeness.

Optimal Healing via Convex Optimization

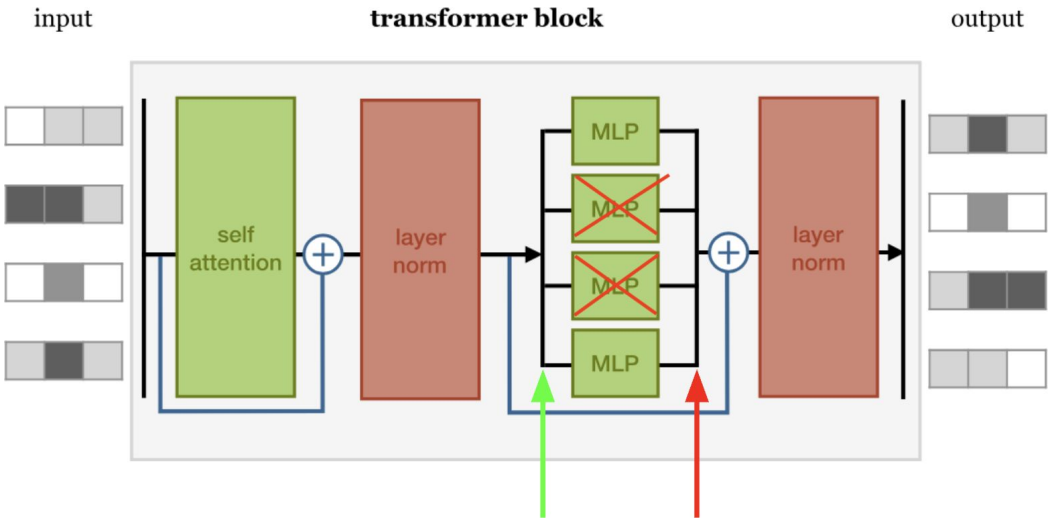
- The set of optimal neural networks with optimal training loss can be characterized
- **Idea:** pick the model with the fewest neurons



[Mishkin & Pilanci ICML 2023; Kim & Mishkin & Pilanci ICLR 2025]

`github.com/pilancilab/scnn`

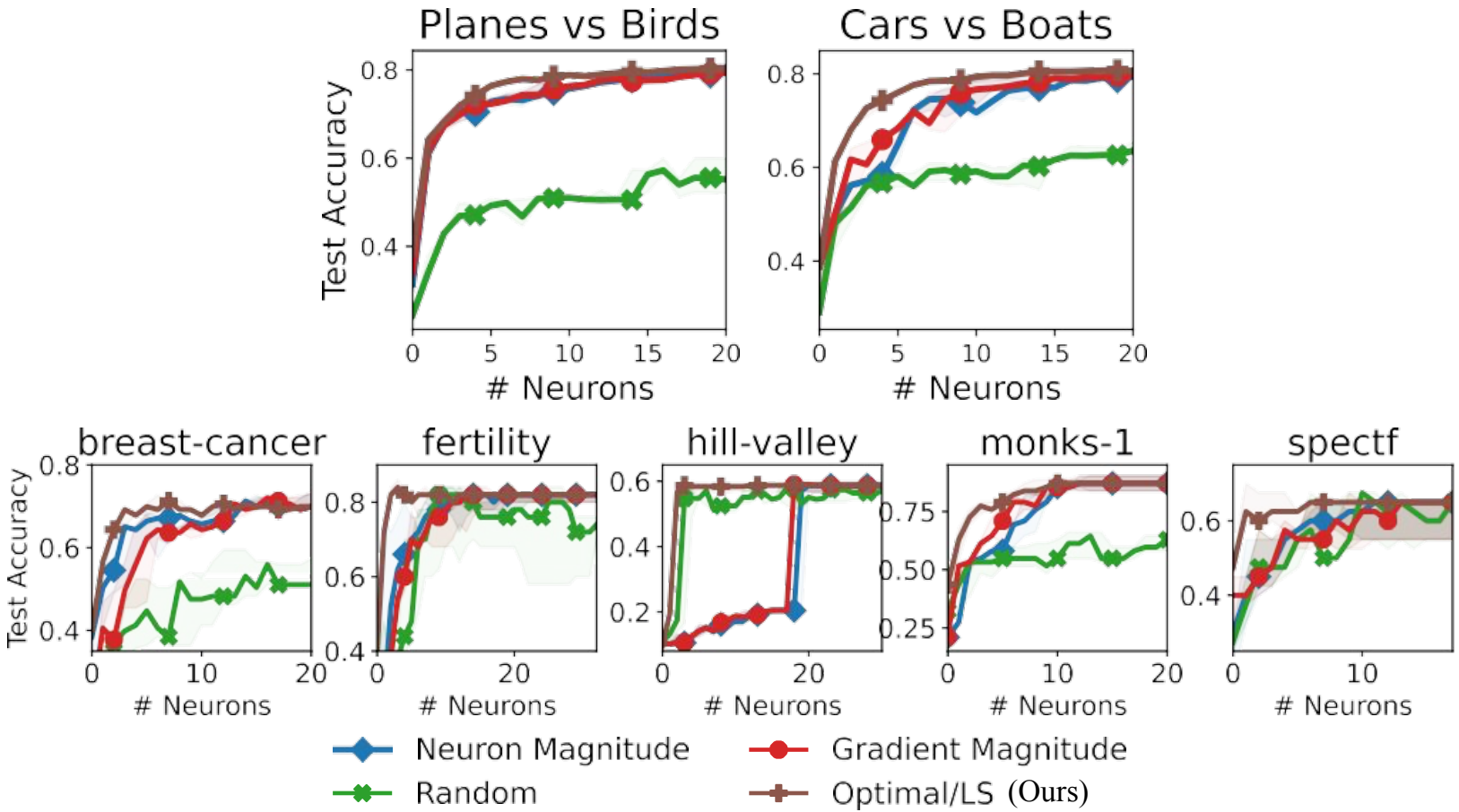
Pruning with Optimal Healing



1. Probe input / output activations of MLP blocks of the original model
2. Prune the units (e.g., channels) using any strategy
3. **Healing:** Re-optimize the remaining units to fit the activations by minimizing least-squares loss

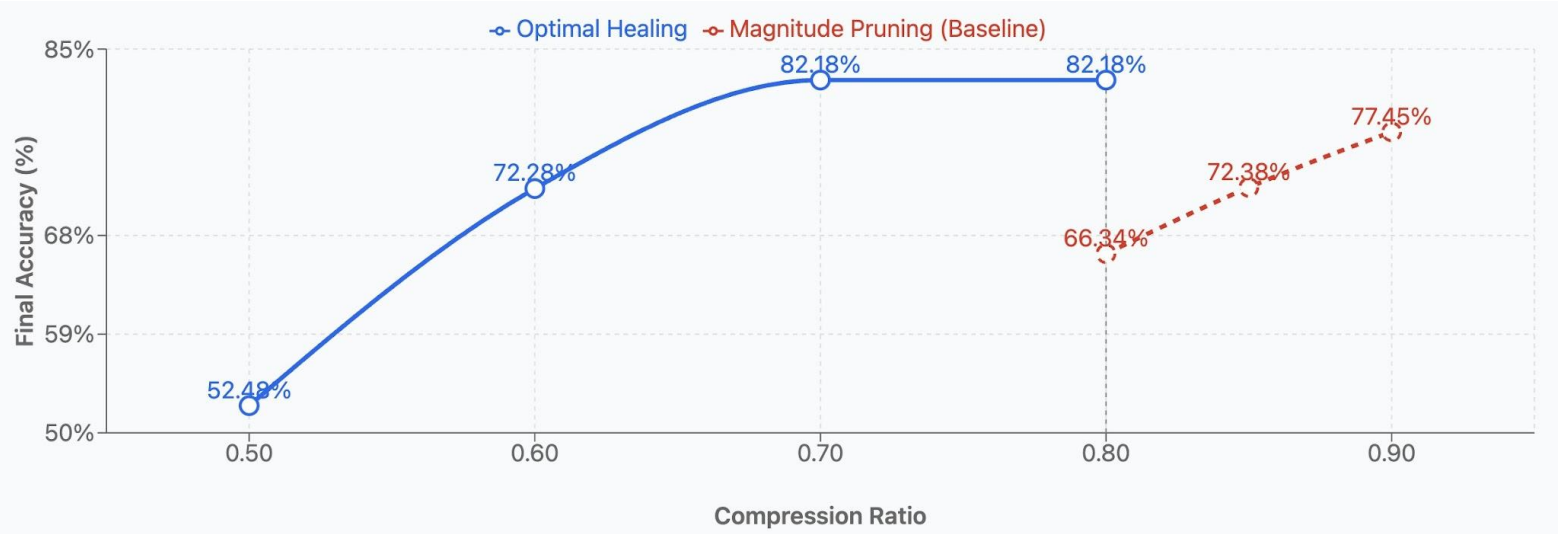
   The healing step requires solving a small local optimization problem for each block

[Mishkin & Pilanci, ICML 2023]



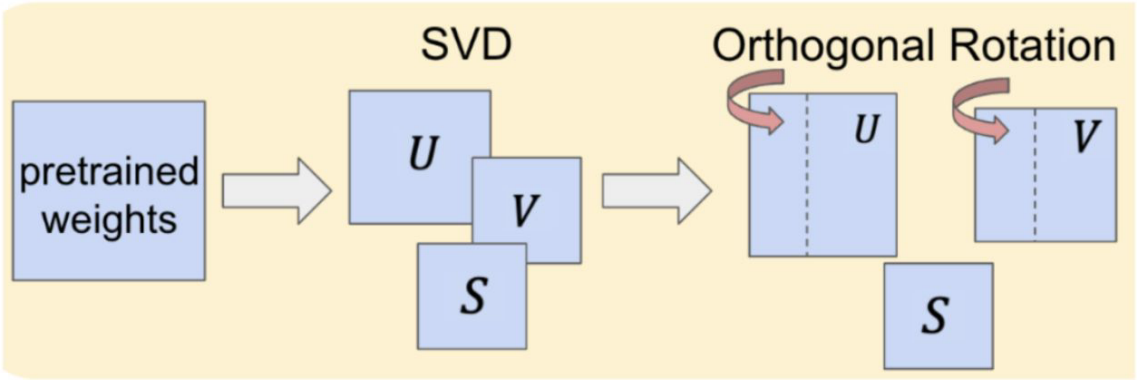
UCI datasets [Mishkin & Sahiner & Pilanci, ICML 2023]

Pruning LLaVA: POPE Benchmark



Mixed-precision Fine-tuning of language models

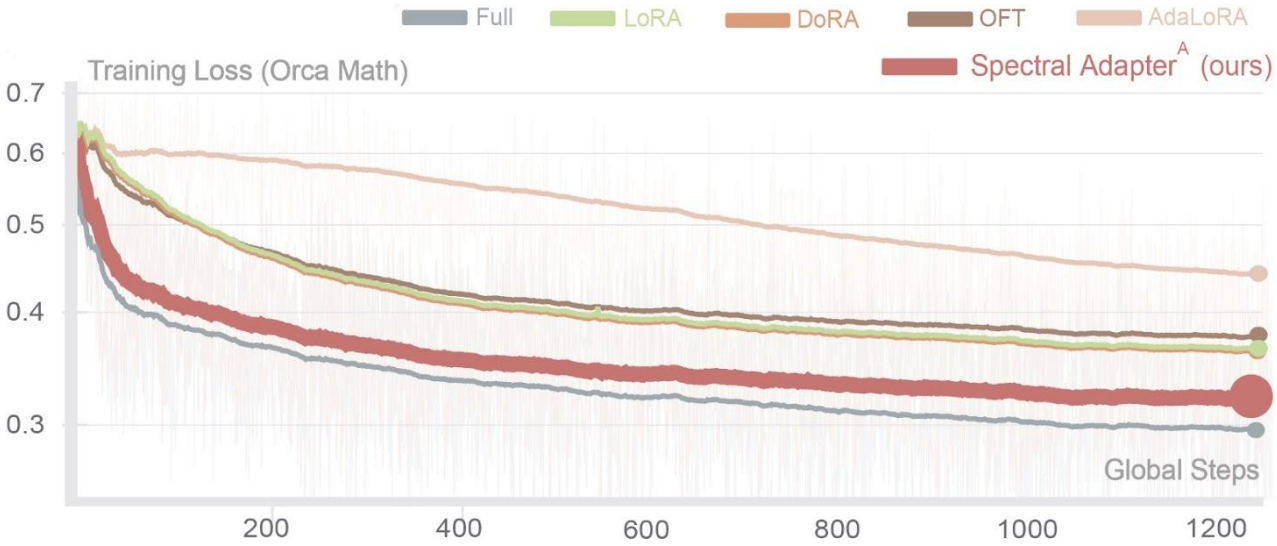
- **LoRA**$(A) = A + LR$ (Low Rank Adapter, Hu et al., 2021)
- **QLoRA**$(A) = Q(A) + LR$ (Quantized Low Rank Adapter, Dettmers et al., 2023)
- our fine-tuning adapter: $\mathrm{SVD}(A) = U_{\text{top}}\Sigma_{\text{top}}V_{\text{top}}^T + U_{\text{bottom}}\Sigma_{\text{bottom}}V_{\text{bottom}}^T$
  fine-tune top-k and quantize bottom-k columns of U and V (same model size as QLORA)

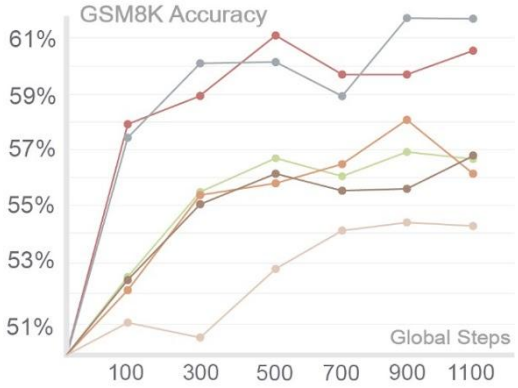


[Zhang, Pilanci, Fine-Tuning in Spectral Space, NeurIPS 2024]

Fine-tuning Large Language Models

- training loss of fine-tuning Llama3 8B model with Orca-Math dataset and accuracy on the GSM8K benchmark





Stable Diffusion (Stability AI), 2022

prompt: A photograph of an astronaut riding a horse

Fine-tuning cont'd

- fine-tuning a diffusion model on some chair images
- associated to text V_{chair}
- after fine-tuning we prompt **a yellow** V_{chair}, **an old** V_{chair} etc.

original

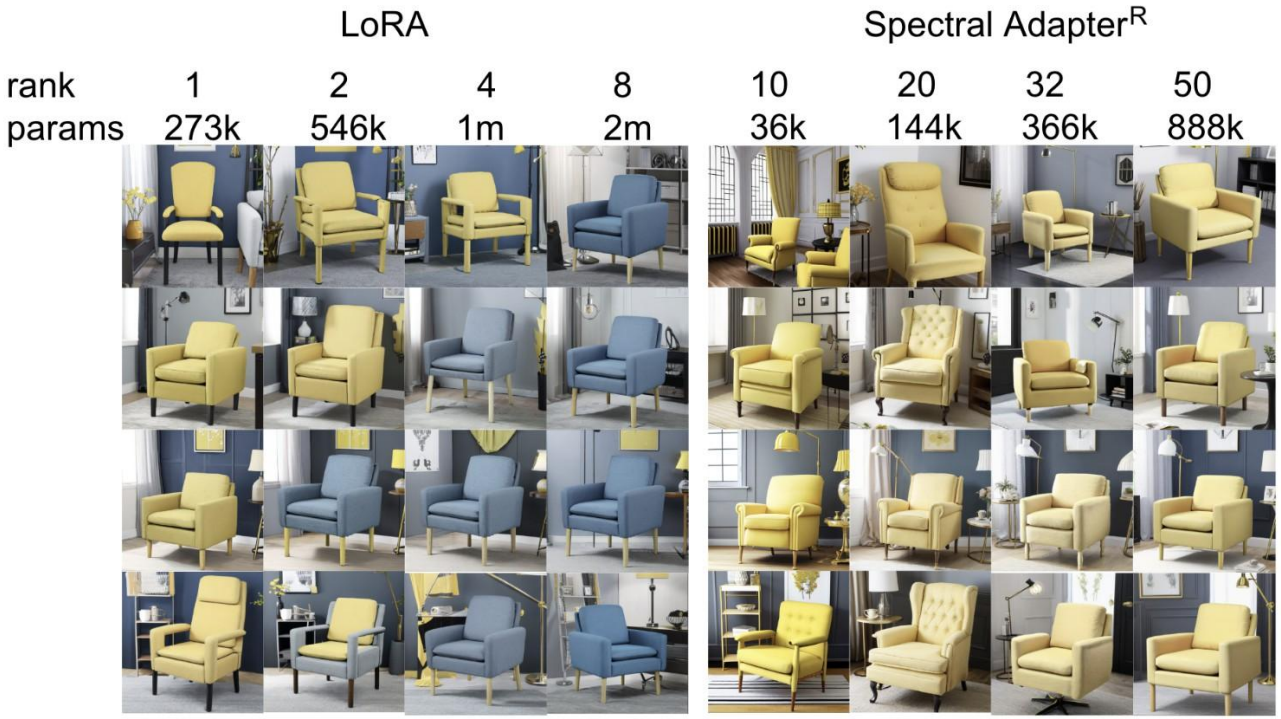
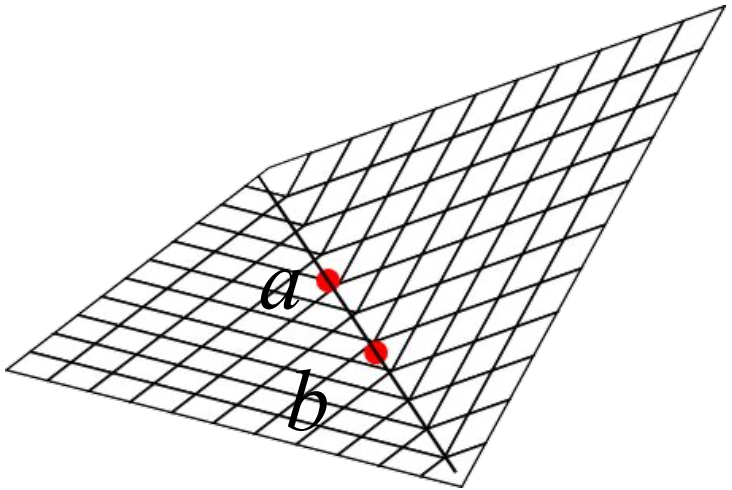


Figure 13: Generation results for prompt "a yellow <V_{chair}>" with various trainable parameter budgets. See Appendix D.3 for experimental details.

Training and fine-tuning: Insights from the theory of deep neural networks

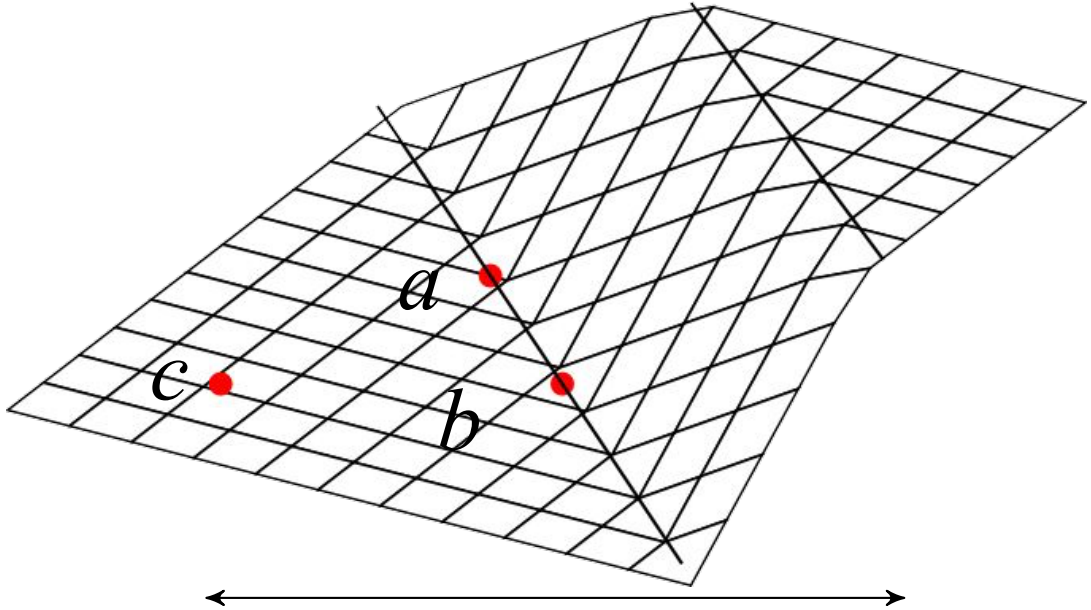


ReLU activation

- like folding a sheet of paper
- **theory:** breakline of each neuron passes through two training samples a and b
- neuron's output is the directional distance to the line

[Pilanci, From Complexity to Clarity..., 2023; Zeger et al. A Library of Mirrors, 2024]

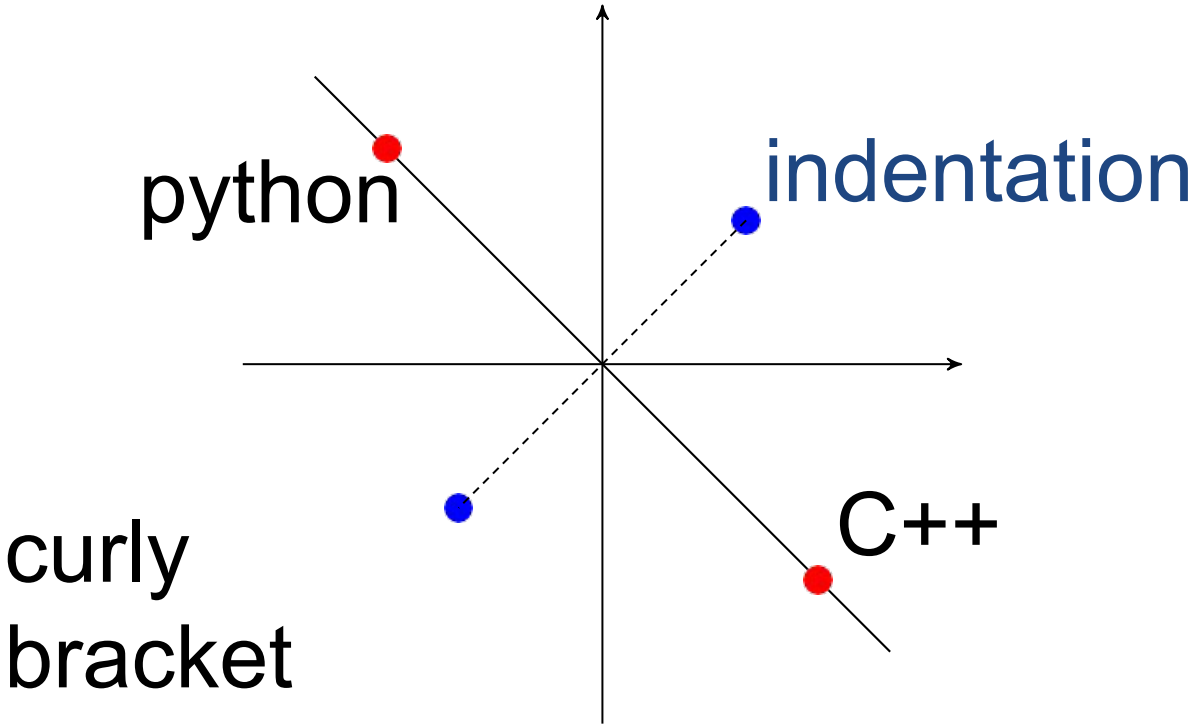
Deeper networks create reflections



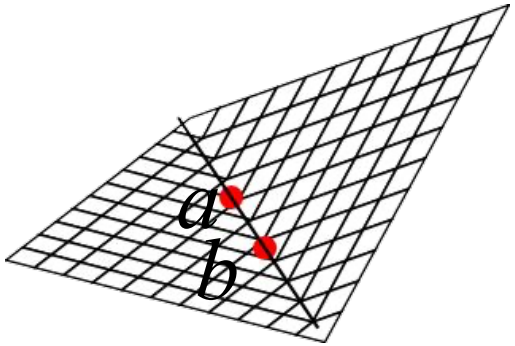
reflection of c around a-b

symmetry axis is the concept 'a **and** b'

Deep networks learn concepts via reflections



Information stored in a deep network



- each neuron measures **directional distance** to a subspace
- **subspaces represen**

  concepts our quantization

  model

  $$Q + LR$$

- the **low-rank** part LR corresponds to these subspaces
- we need higher precision for L and R

References & Further Reading

- Ang, C. et al. “Optimal Scalar Quantization for Matrix Multiplication: Closed-Form Density and Phase Transition” , 2026
- M. Erdogan et al. “Linearizing Pretrained LLMs via Gated Delta Attention Distillation”, 2026
- Saha et al. CALDERA, 2024
- Kahan, W. “Paranoia: A Floating-Point Benchmark.” *Byte Magazine*, 10(2), Feb 1985.
- GAO Report IMTEC-92-26, “Patriot Missile Defense: Software Problem Led to System Failure at Dhahran.” 1992.
- Shirriff, K. “Architecture of the Apollo Guidance Computer.” righto.com, 2019.
- Quinn, K. “Ever Had Problems Rounding Off Figures?” *Wall Street Journal*, Nov 8, 1983.
- Frantar, E. et al. “GPTQ: Accurate Post-Training Quantization for GPT.” ICLR 2023.
- Lin, J. et al. “AWQ: Activation-aware Weight Quantization.” MLSys 2024.
- Dettmers, T. et al. “QLoRA: Efficient Finetuning of Quantized LLMs.” NeurIPS 2023.
- Ma, S. et al. “The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits.” 2024.
- Chee, J. et al. “QuIP: 2-Bit Quantization of LLMs With Guarantees.” NeurIPS 2023.
- Rouhani, B. D. et al. “Microscaling Data Formats for Deep Learning.” arXiv:2310.10537, 2023.
- OCP Microscaling Formats (MX) Specification v1.0. Open Compute Project, Sep 2023.
- NVIDIA. “Introducing NVFP4 for Efficient and Accurate Low-Precision Inference.” NVIDIA Blog, 2025.
- Egiazarian, V. et al. “Bridging the Gap Between Promise and Performance for Microscaling FP4 Quantization (MR-GPTQ).” 2025.
- Chen, X. et al. “TetraJet-v2: Accurate NVFP4 Training for LLMs.” 2025.
- Li, Y. et al. “SVDQuant Meets NVFP4.” MIT Han Lab, 2025.

Summary, Future Work and Acknowledgements

- Bits are expensive again: Quantization is critical
  - task-aware quantizers and low-rank factorizations
- Transformer efficiency stack is effective
  - Quantization
  - Pruning
  - Distillation
  - Fine-tuning
- **Future work:** co-design of model architecture and compute kernels
- **Acknowledgements:**
  - **Amazon Research Grant (Mentor: Rahul Bakshi)**
  - **Apple**