verda
formerly DataCrunch

EurIPS

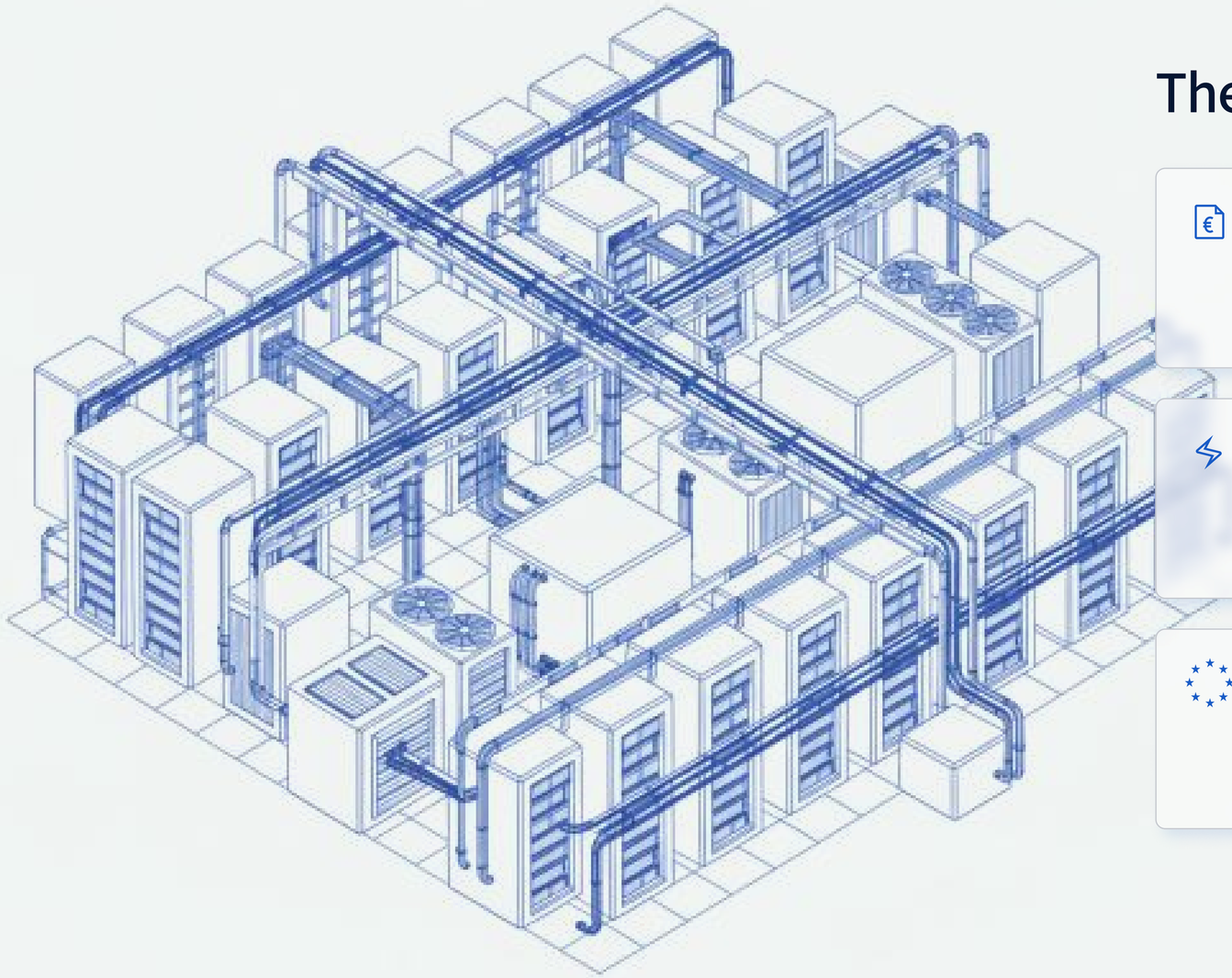
Unlocking FP4: Low Precision AI on NVIDIA Blackwell

Speakers

Paul Chang
ML Engineer

Riccardo Mereu
ML Engineer

Hyperscalers were built for CPU - AI cloud is different



The AI Cloud

Compute cost matters

Legacy cloud was built around saving developer time.
AI cloud optimizes for performance and compute costs.

GPU Data Centers need power

Data center build out starts from zero as energy density, power supply and cooling innovations make old footprint obsolete

Sovereignty is critical

Geopolitical shift towards national sovereignty especially in Europe requires an AI-native Hyperscaler built in Europe

Our AI-Native cloud beats hyperscalers for AI workloads

Clean modern architecture

Redesigned from scratch for GPU workloads, beating legacy piecemeal cloud architecture

Unified control plane

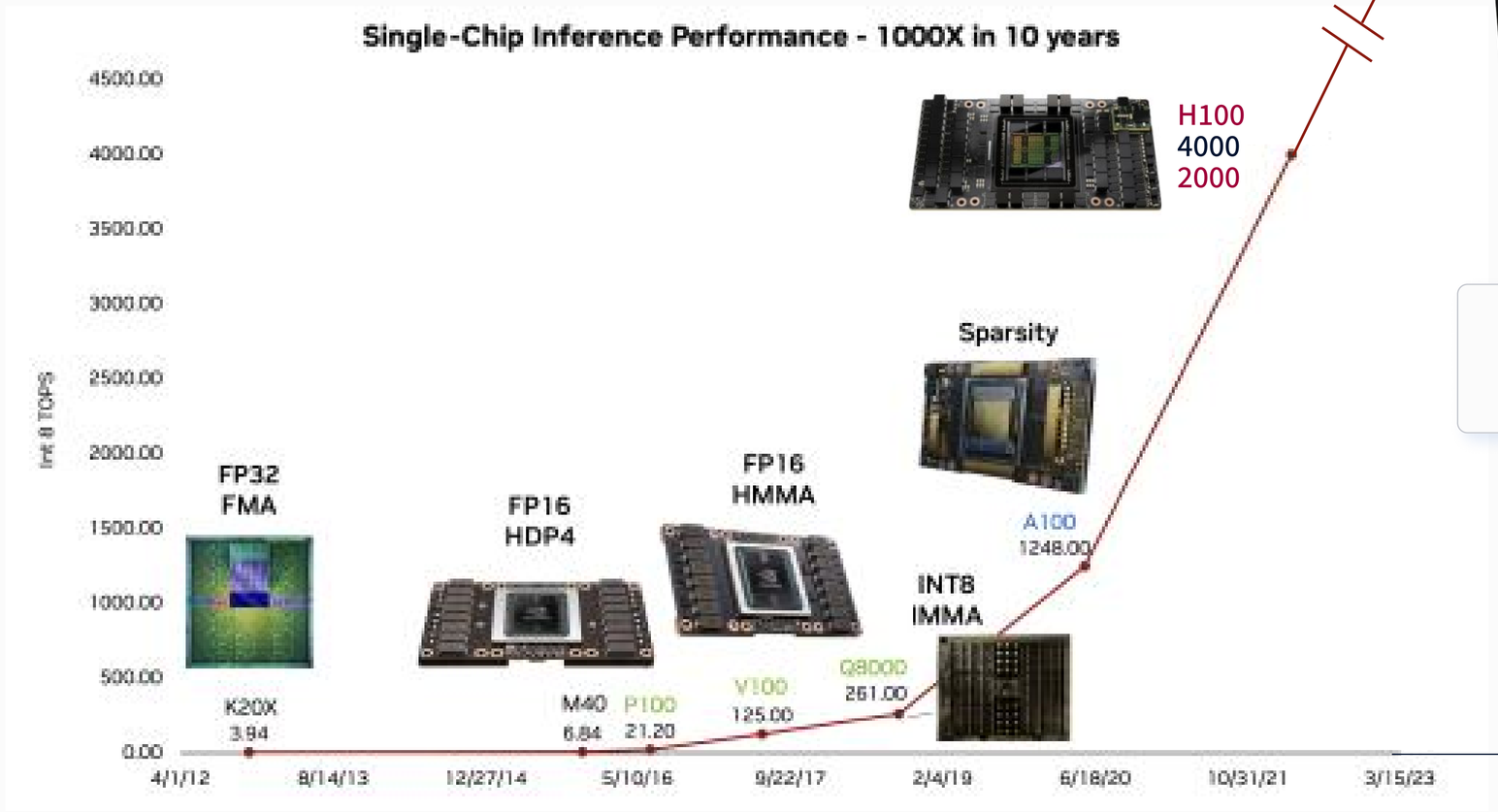
One cohesive compute layer, beating legacy fragmented layer system on utilisation and developer experience

Deep AI framework integration

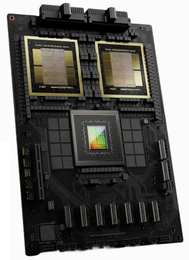
Natively optimised for PyTorch, JAX and more, enabling faster adoption of new AI tooling

How compute has gone 1000x?

verda



B300
FP4 14000



B200
FP4 9000
FP8 4500

NVL72

Or 3500x increase in 15 year?

Source: Hardware for Deep Learning, Bill Dally

Where does the increases come from?

verda

- Number Representation:
  16x (64x) FP32 → BF16 → FP8 → FP4.

- Amoritized Instructions (Tensor cores):
  12.5x FMA → HMMA → WGMMA → UMMA

- Process (fab size):
  2.5x 28nm → 16nm → 7nm → 5nm (more transistors per mm^2)

- Model efficiency has also boosted performance, FA, FA2, FA3, FA4.

Section

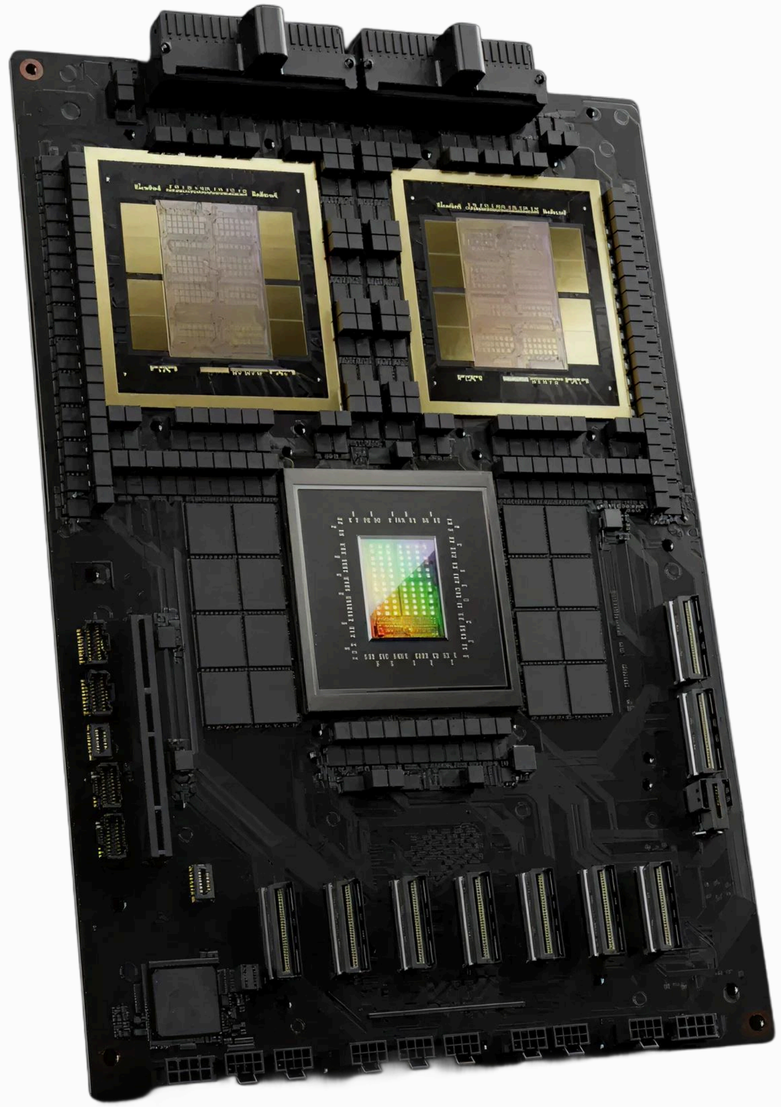
Low Precision AI

Unlocking FP4:
Low Precision AI on NVIDIA Blackwell

verda

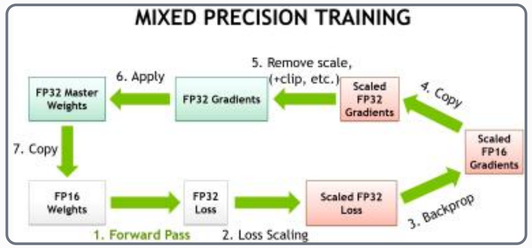
Outline

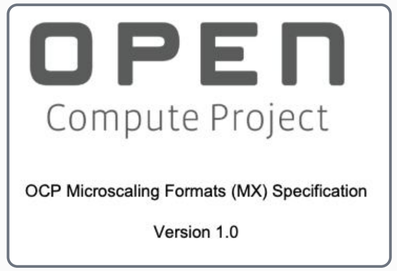
1. Why do we need Low Precision AI?
2. What is NVFP4?
3. How NVIDIA Blackwell enable this?



verda

Timeline: Floating Point in AI





NVIDIA
Pre-Volta Arch.

2012–2015

Early DL models
(DanNet, AlexNet, ...)
CUDA - FP32

Caffe/Theano/TF/
NVIDIA Apex
FP16

2016

NVIDIA
Tesla V100

2017

Google *TPUv2*
BF16

NVIDIA
Ampere A100
(native BF16)

2020

2022

NVIDIA
Hopper HX00 - **FP8**

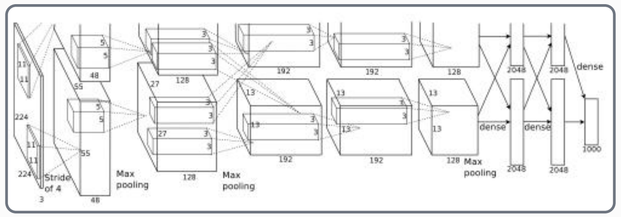
OCP Microscaling
MXFP6 - MXFP4

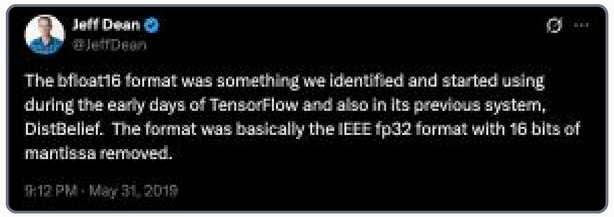
2023

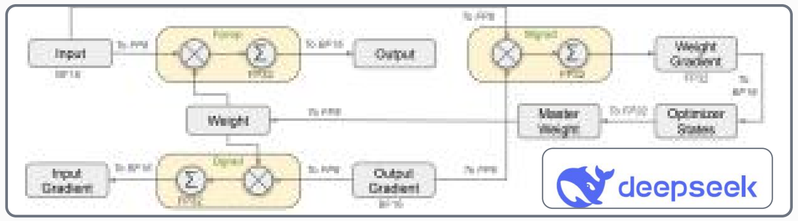
NVIDIA
Blackwell
(BX00 and GBX00)
NVFP4

2024

DeepSeek
(and frontier labs)
FP8 QAT





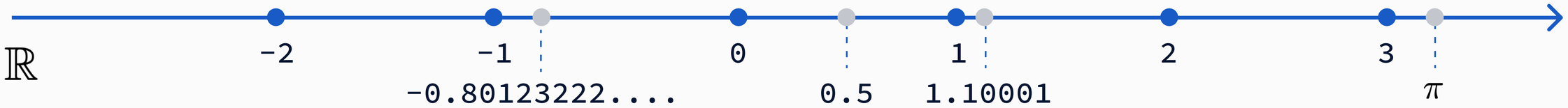


verda

Floating Point Basics

Real Number Line:

- Arbitrarily large and small values → **Infinite range**
- Also between any pair of numbers → **Infinite precision**



We cannot do infinite:

- Finite memory
- Finite silicon area (or time to complete operations)

Need for number representation to sample the real line:

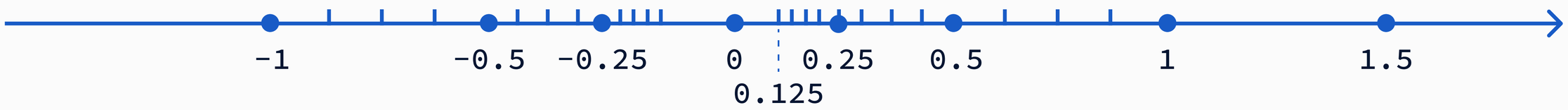
- Integer variants
- Floating Point

verda

Floating Point Basics

FP Line:

- FP samples the real number line with equal number of samples between each pair of adjacent powers of 2
- *Example: E5M10 has 1024 samples between 0.25 and 0.5, between 128 and 256*



Bit Fields (ExMy):

- **Sign:** 1 bit (0 positive, 1 negative)
- **Exponent:** E bits, which power of 2 is sampled → *dynamic range*
- **Mantissa:** M bits, samples between powers of two → *precision*

$$N_{10} = (-1)^S \times 1.M_{10} \times 2^{E_{10} - \text{bias}}$$

Exponent bias = $2^{E_N - 1} - 1$

verda

Floating Point Basics

FP16 Example (E5M10)

- **Sign:** 1 bit (0 positive, 1 negative)
- **Exponent:** 5 bits $2^{5-1} - 1 = 15$
- **Mantissa:** 10 bits
- **Exponent bias:**

$$N_{10} = (-1)^S \times 1.M_{10} \times 2^{E_{10} - \text{bias}}$$

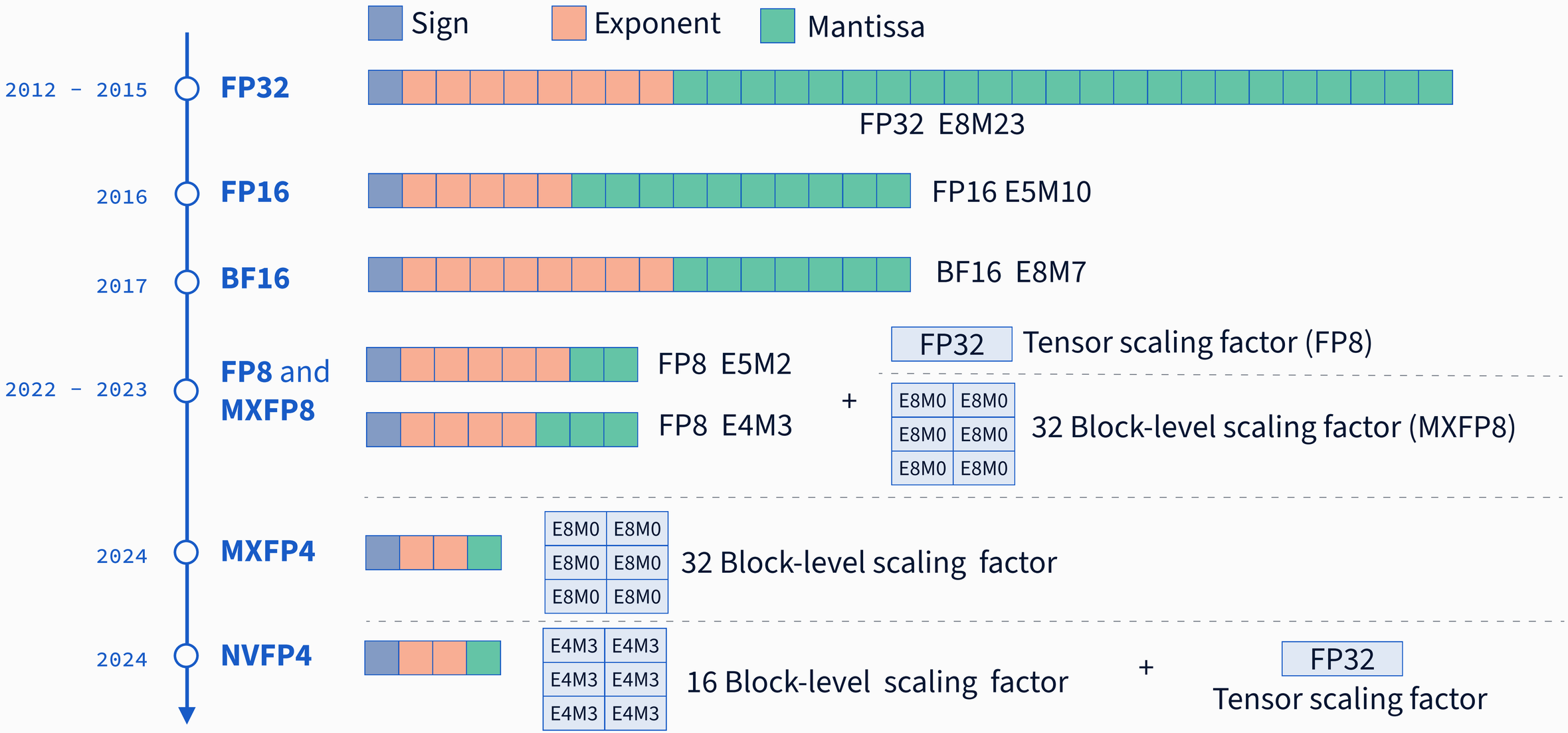
FP 16 bits:

0.10001.1010000000 = $(6.510)_{10}$

$$(-1)^0 \times 1.101_2 \times 2^{10001_2 - 15} = (1 \times (1 + 2^{-1} + 2^{-3}) \times 2^2)_{10} =$$
$$= (4 \times 1.62510)_{10} = (6.510)_{10}$$

verda

Floating Point in AI



verda

Model Quantization Algorithms

There are several strategies to quantize neural networks:

- **Post Training Quantization (PTQ)**
  - Turn pre-trained ***weights*** *and/or* ***activations*** into low-precision formats
  - No need for training data or training hardware
  - May lead to a lower quality model

Neural Network Inference

Slides inspired by GPU Mode Lec. 30:
https://www.youtube.com/@GPUMODE

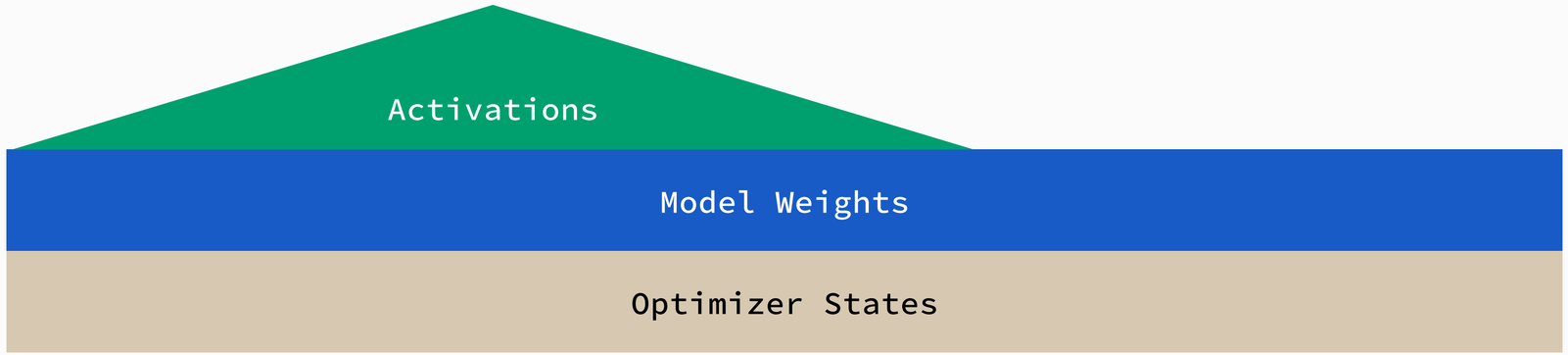
verda

Model Quantization Algorithms

There are several strategies to quantize neural networks:

- **Post Training Quantization (PTQ)**
  - Turn pre-trained ***weights*** *and/or* ***activations*** into low-precision formats
  - No need for training data or training hardware
  - May lead to a lower quality model
- **Quantization Aware Training (QAT)**
  - Adds quantization to the forward for training/fine-tuning

Neural Network FWD

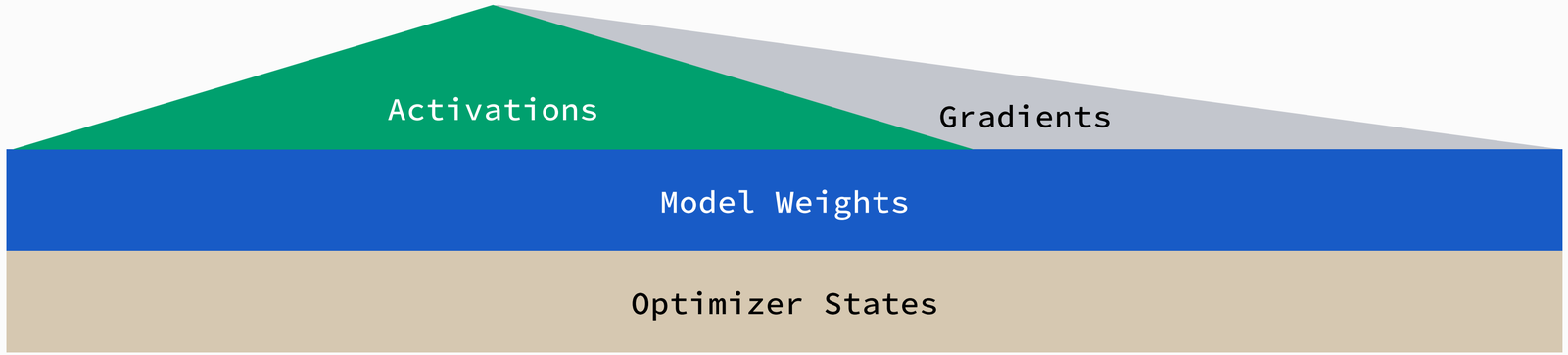


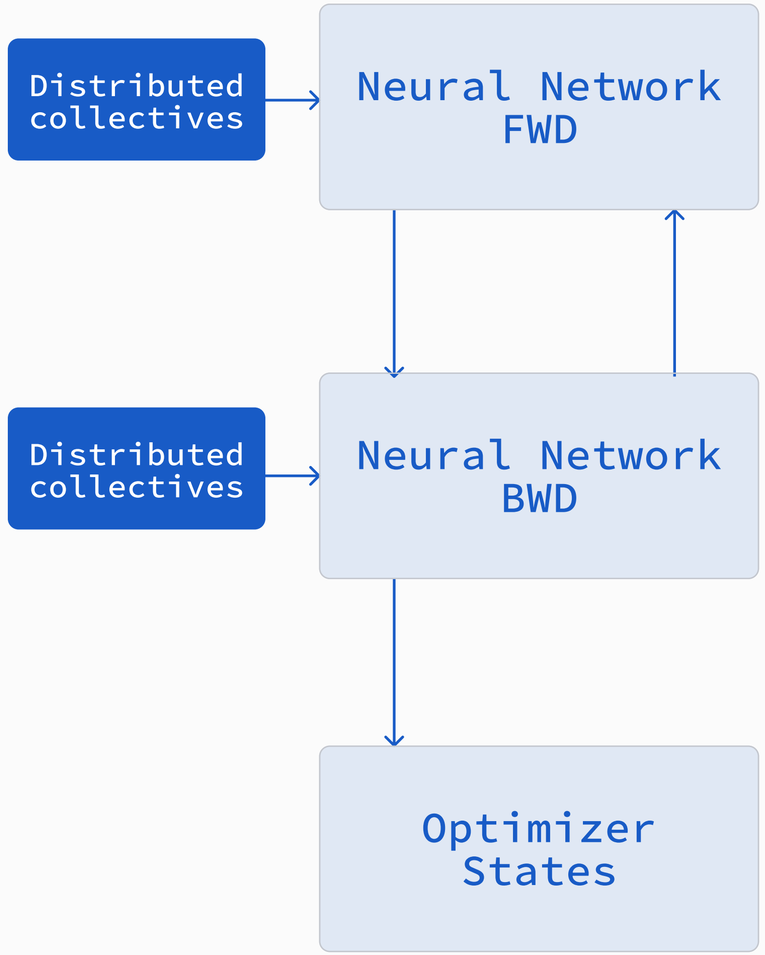
verda

Model Quantization Algorithms

There are several strategies to quantize neural networks:

- **Post Training Quantization (PTQ)**
  - Turn pre-trained ***weights*** *and/or* ***activations*** into low-precision formats
  - No need for training data or training hardware
  - May lead to a lower quality model
- **Quantization Aware Training (QAT)**
  - Adds quantization to the forward for training/fine-tuning
- **Quantized Training (QT)**
  - Adds quantization also to the backward pass improving over QAT





verda

DeepSeek-V3 Mixed Precision

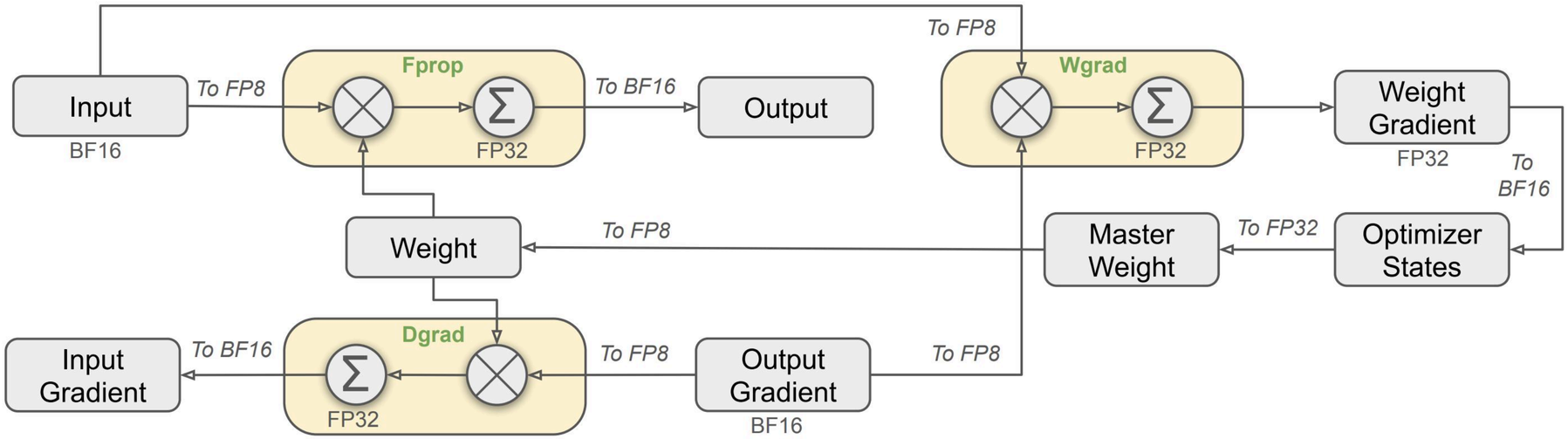
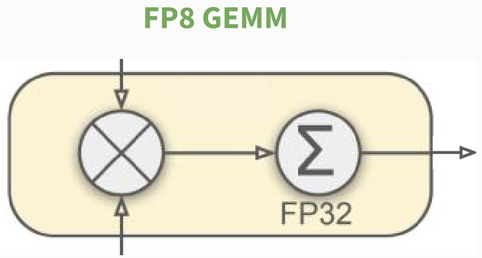


Figure 6 | The overall mixed precision framework with FP8 data format. For clarification, only the `Linear` operator is illustrated.



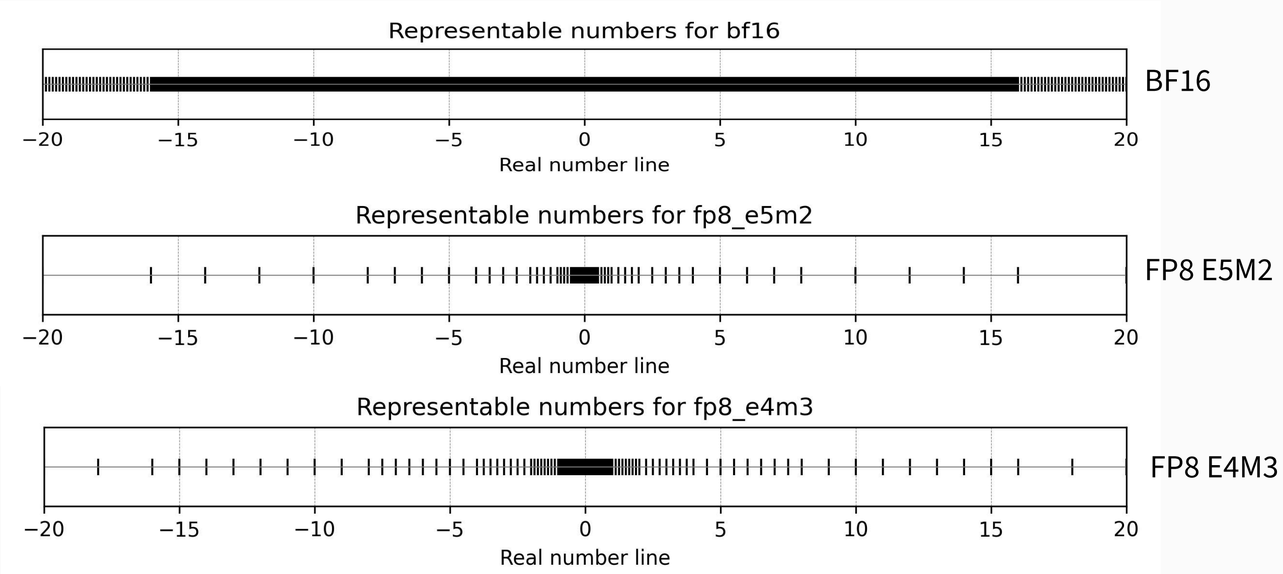
Source: DeepSeekv3 Technical Report, https://arxiv.org/abs/2412.19437v2

NVFP4

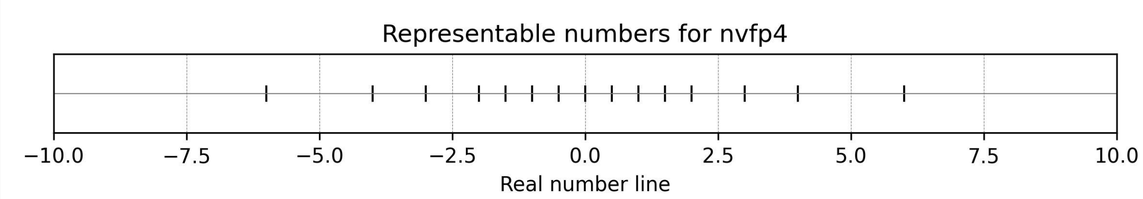
verda

NVFP4: Are 16 values enough?

sign	exponent	mantissa		sign	exponent	mantissa	
0	0	0	0 = 0.0	1	0	0	0 = -0.0
0	0	0	1 = 0.5	1	0	0	1 = -0.5
0	0	1	0 = 1.0	1	0	1	0 = -1.0
0	0	1	1 = 1.5	1	0	1	1 = -1.5
0	1	0	0 = 2.0	1	1	0	0 = -2.0
0	1	0	1 = 3.0	1	1	0	1 = -3.0
0	1	1	0 = 4.0	1	1	1	0 = -4.0
0	1	1	1 = 6.0	1	1	1	1 = -6.0



NVFP4 E2M1

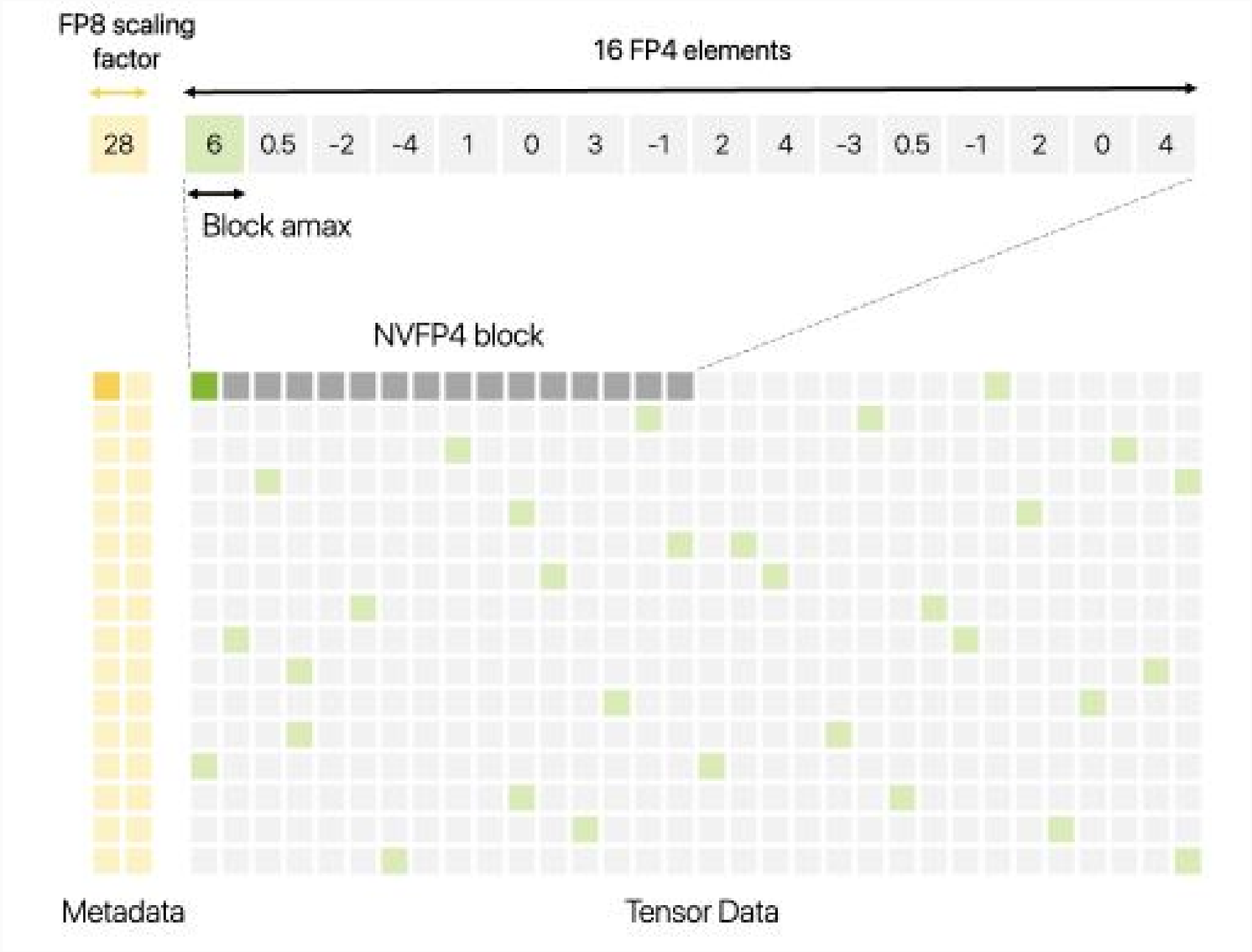


verda

NVFP4 vs. MXFP4

The smaller block size and more precise scaling allows:

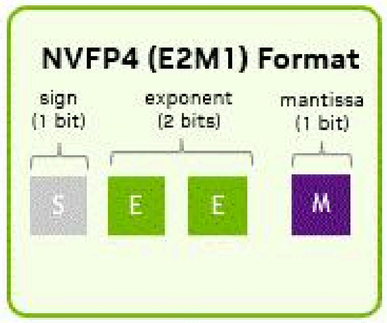
- to increase accuracy of outliers
- to minimize the values quantized to zero



Source: Pretraining Large Language Models with NVFP4 – arxiv:2509.25149

NVFP4

NVIDIA NVFP4



Source: Pretraining Large Language Models with NVFP4 - arxiv:2509.25149

verda

NVFP4 Recipe

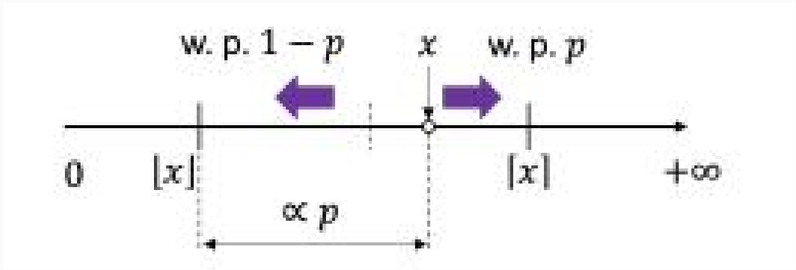
NVFP4 Recipe:

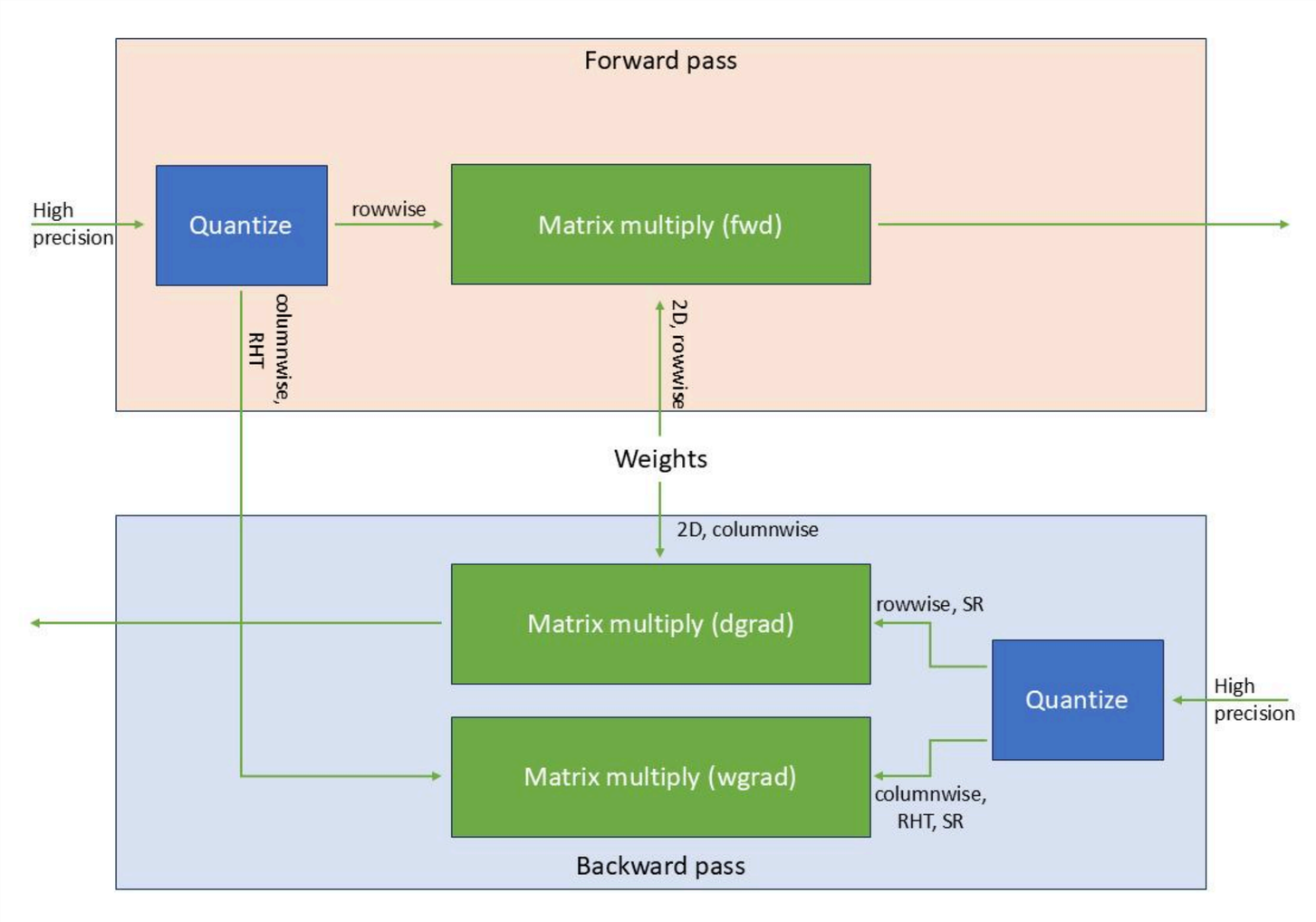
- 2D Scaling
- Stochastic Rounding

$$\mathrm{Round}(x) = \begin{cases} \lfloor x \rfloor, & \text{w/ prob. } 1-p, \\ \lceil x \rceil, & \text{w/ prob. } p. \end{cases}$$

$$p = (x - \lfloor x \rfloor)/(\lceil x \rceil - \lfloor x \rfloor)$$

$$\mathbb{E}[\mathrm{Round}(x)] = x$$





Linear Layer flow w/ NVFP4

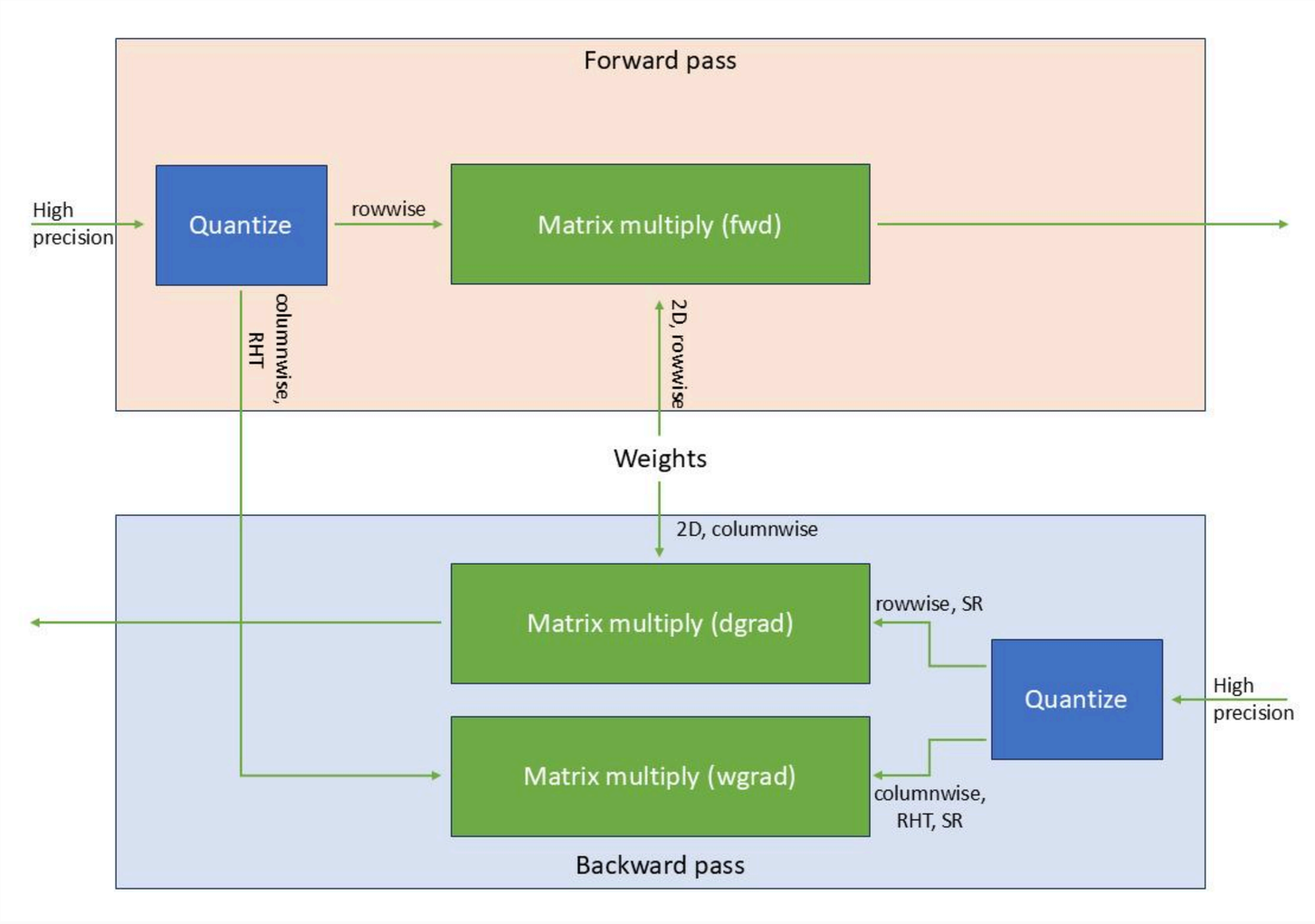
Source: Pretraining Large Language Models with NVFP4 - arxiv:2509.25149

verda

NVFP4 Recipe

NVFP4 Recipe:

- 2D Scaling
- Stochastic Rounding
- Random Hadamard Transforms

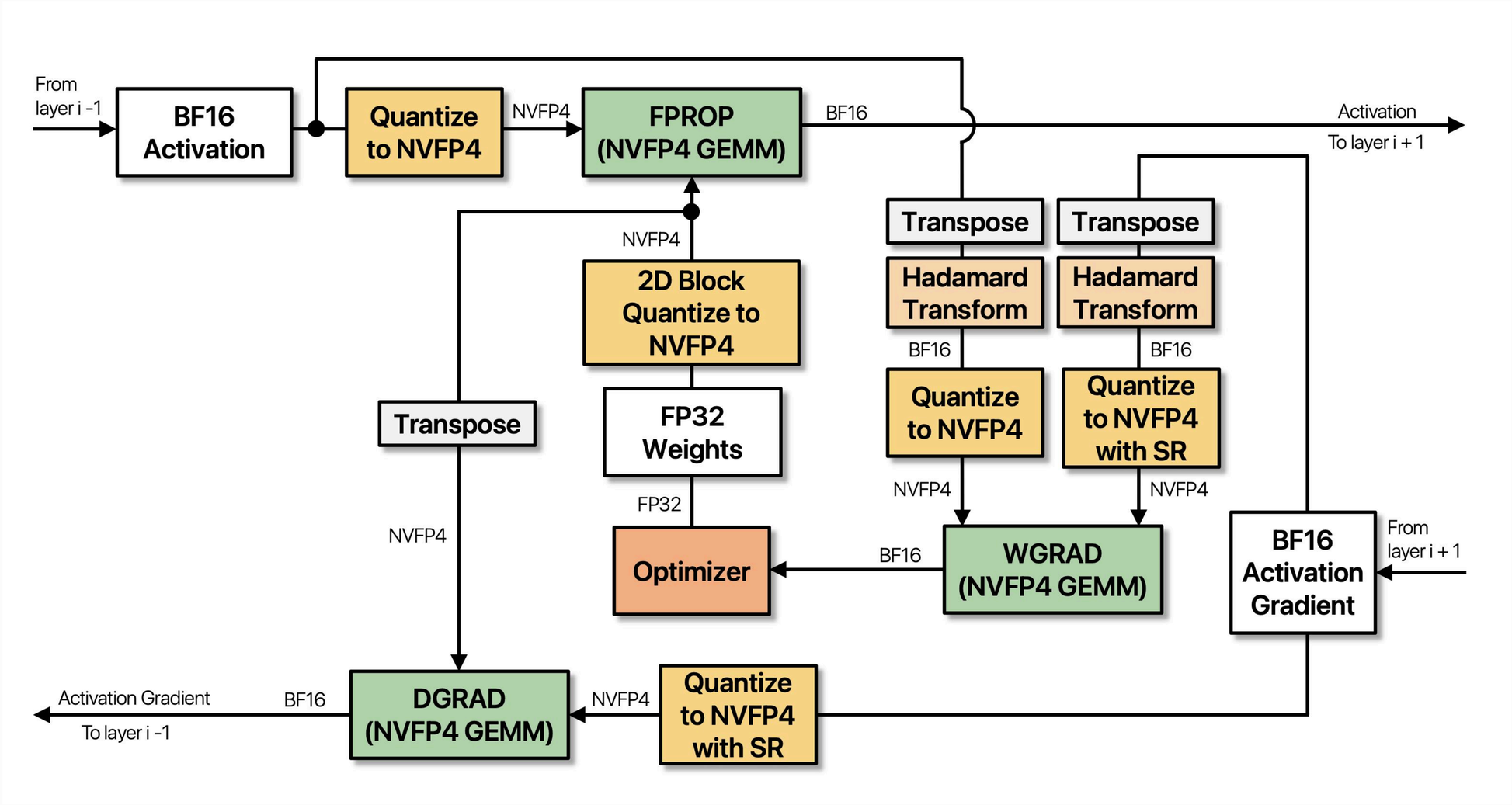


Linear Layer flow w/ NVFP4

Source: Pretraining Large Language Models with NVFP4 – arxiv:2509.25149

verda

NVIDIA NVFP4 Recipe



Source: Pretraining Large Language Models with NVFP4 – arxiv:2509.25149

verda
formerly DataCrunch

Thanks for listening!

Use the code:

EURIPS-2025

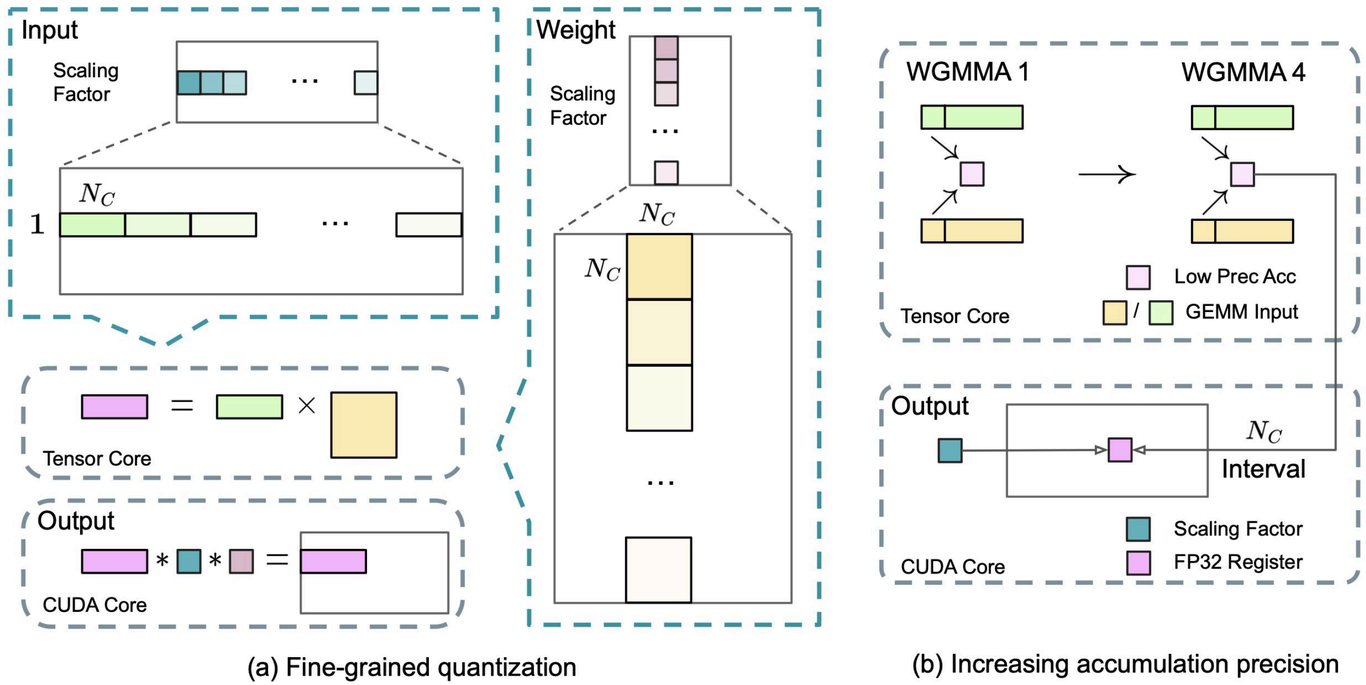
(50€ credit)

Contacts:

Paul Chang
ML Engineer

Riccardo Mereu
ML Engineer

DeepSeek V3



(a) Fine-grained quantization

(b) Increasing accumulation precision

Figure 7 | (a) We propose a fine-grained quantization method to mitigate quantization errors caused by feature outliers; for illustration simplicity, only `Fprop` is illustrated. (b) In conjunction with our quantization strategy, we improve the FP8 GEMM precision by promoting to CUDA Cores at an interval of $N_C = 128$ elements MMA for the high-precision accumulation.

Source: DeepSeekv3 Technical Report, https://arxiv.org/abs/2412.19437v2