

LLM Sys

LLM Quantization -- Basic methods

Lei Li



Language
Technologies
Institute

Carnegie Mellon University
School of Computer Science

Ack: Kath Choi, Aashiq Muhamed, Zhen Wu, Hongyi Jin, Wayne Wang

Today's Topic

- Low precision numbers in computer
- Basic Quantization Methods

LLM Training and Inference are Costly!

Llama-70B

39.3M H100-80GB GPU hours to train
requires 140GB GPU memory for inference

Deepseek V3 (671GB)

2.8M H800 GPU hours to train
requires > 400GB GPU memory for inference

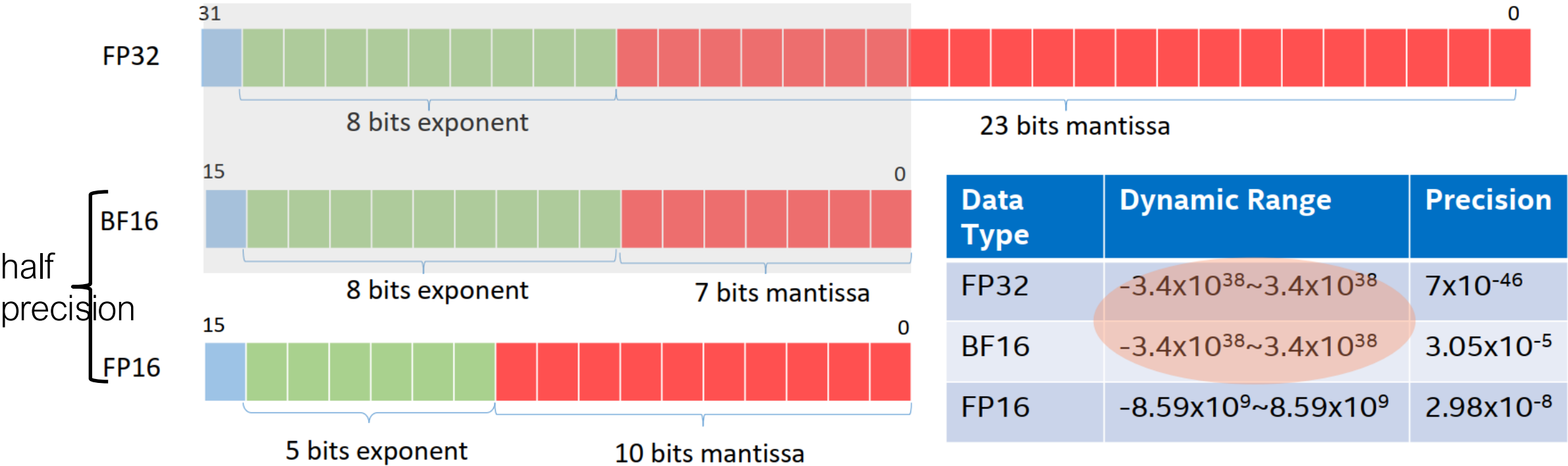


Model Quantization

- Use low-bit precision to store parameters and layer outputs
- Quantization can
 - reduce memory → larger batch size
 - speed up calculation, more operations in one cycle
- Cons: potentially reduce accuracy

Precision Formats

$$\pm 1.(\textit{mantissa}) * 2^{\{a-2^{\{\textit{exponent}-1\}}\}}$$



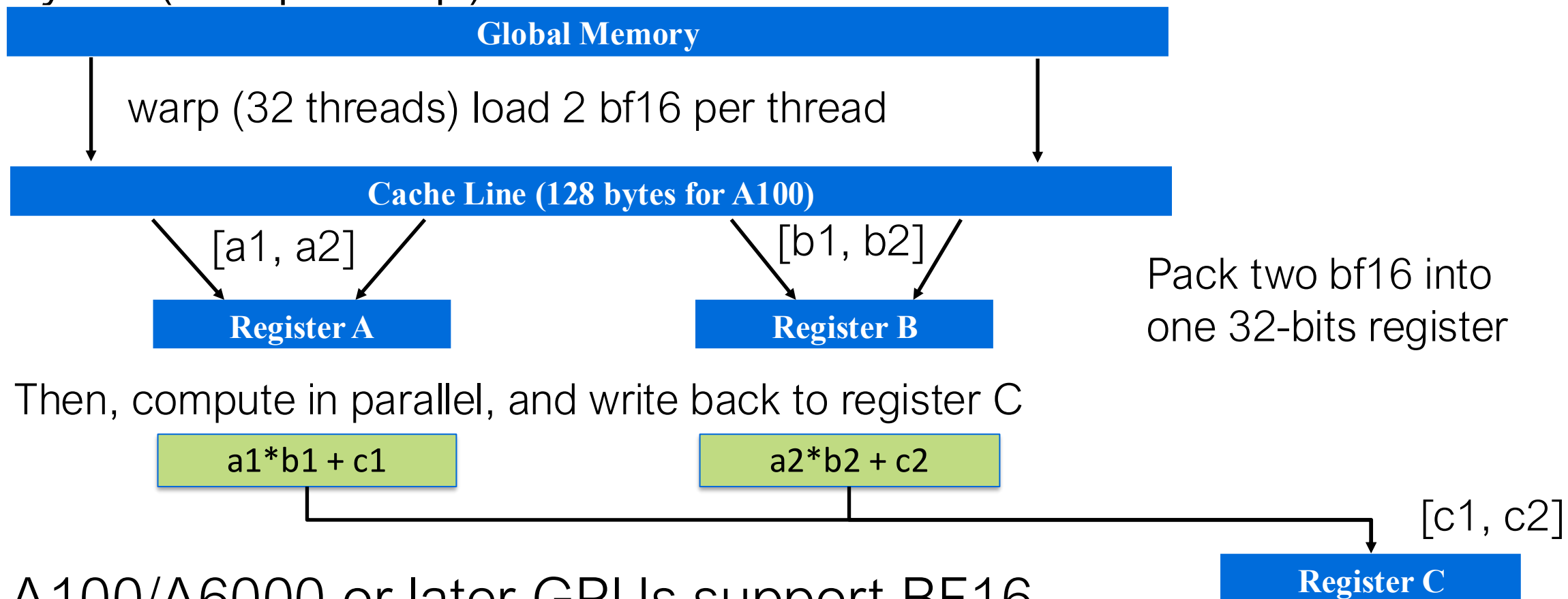
BF16 favors Dynamic Range than Precision.

INT8 range: $-128 \sim 127$, Precision: integer



BF16/FP16 Calculations are faster!

HFMA2: Half-precision Fused Multiply-Add for 2 elements in one cycle (2x speedup)



A100/A6000 or later GPUs support BF16

CUDA APIs for Half Precision

Performs half2 vector addition in round-to-nearest-even mode.

```
__device__ __half2 __hadd2(const __half2 a, const __half2 b)
```

Performs half2 vector fused multiply-add in round-to-nearest-even mode.

```
__device__ __half2 __hfma2(const __half2 a, const __half2 b, const __half2 c)
```

Direct Quantization Approach

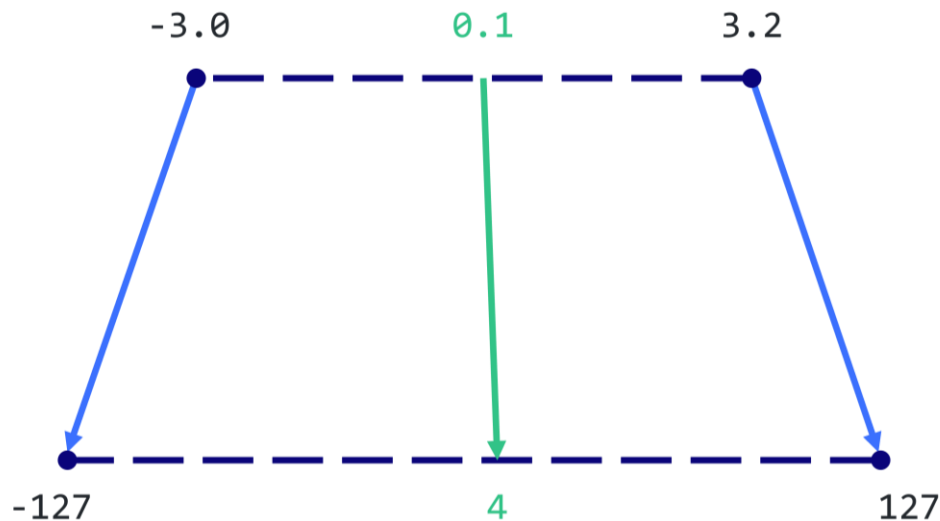
- Using lower precision
 - converting parameters from FP32 to INT8 or INT4
 - perform all computation in lower precision.
- Reduce model accuracy:
 - Loss of Precision → accumulate quantization noise
 - Range mismatch → values are clipped and lead to information loss
 - Quantization error → rounding errors

Quantize a number

- Absmax quant

$$\mathbf{X}_{\text{quant}} = \text{round} \left(\frac{127}{\max |\mathbf{X}|} \cdot \mathbf{X} \right)$$

$$\mathbf{X}_{\text{dequant}} = \frac{\max |\mathbf{X}|}{127} \cdot \mathbf{X}_{\text{quant}}$$



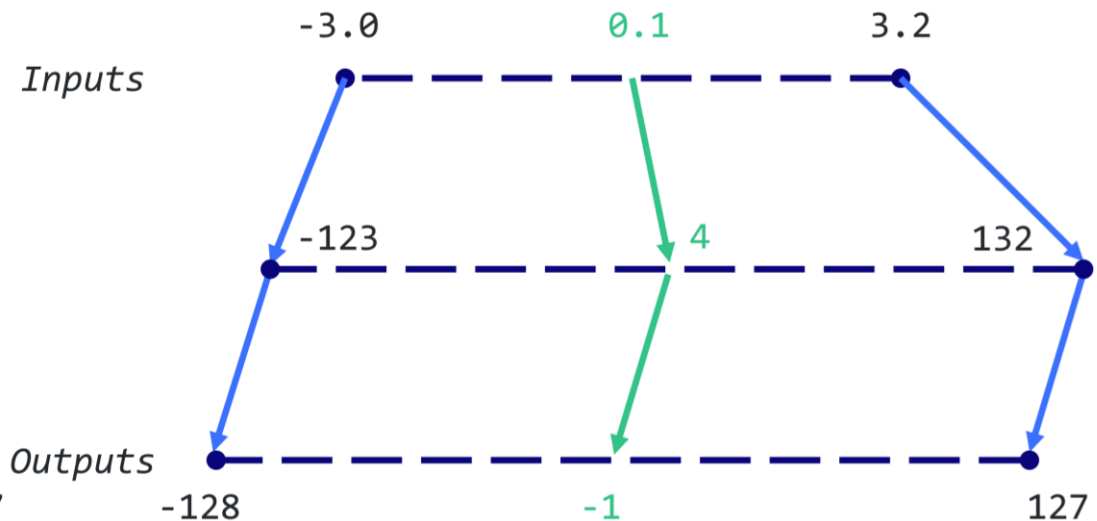
- Zero-point quant

$$\text{scale} = \frac{255}{\max(\mathbf{X}) - \min(\mathbf{X})}$$

$$\text{zeropoint} = -\text{round}(\text{scale} \cdot \min(\mathbf{X})) - 128$$

$$\mathbf{X}_{\text{quant}} = \text{round} \left(\text{scale} \cdot \mathbf{X} + \text{zeropoint} \right)$$

$$\mathbf{X}_{\text{dequant}} = \frac{\mathbf{X}_{\text{quant}} - \text{zeropoint}}{\text{scale}}$$



Code implementation for quant(.)

```
import torch
def absmax_quantize(X):
    # Calculate scale
    scale = 127 / torch.max(torch.abs(X))
    # Quantize
    X_quant = (scale * X).round()
    # Dequantize
    X_dequant = X_quant / scale
    return X_quant.to(torch.int8), X_dequant
```

```
def zeropoint_quantize(X):
    # Calculate value range (denominator)
    x_range = torch.max(X) - torch.min(X)
    x_range = 1 if x_range == 0 else x_range
    # Calculate scale
    scale = 255 / x_range
    # Shift by zero-point
    zeropoint = (-scale * torch.min(X) - 128).round()
    # Scale and round the inputs
    X_quant = torch.clip((X * scale + zeropoint).round(), -128,
127)
    # Dequantize
    X_dequant = (X_quant - zeropoint) / scale
    return X_quant.to(torch.int8), X_dequant
```

Direct Quantization Colab

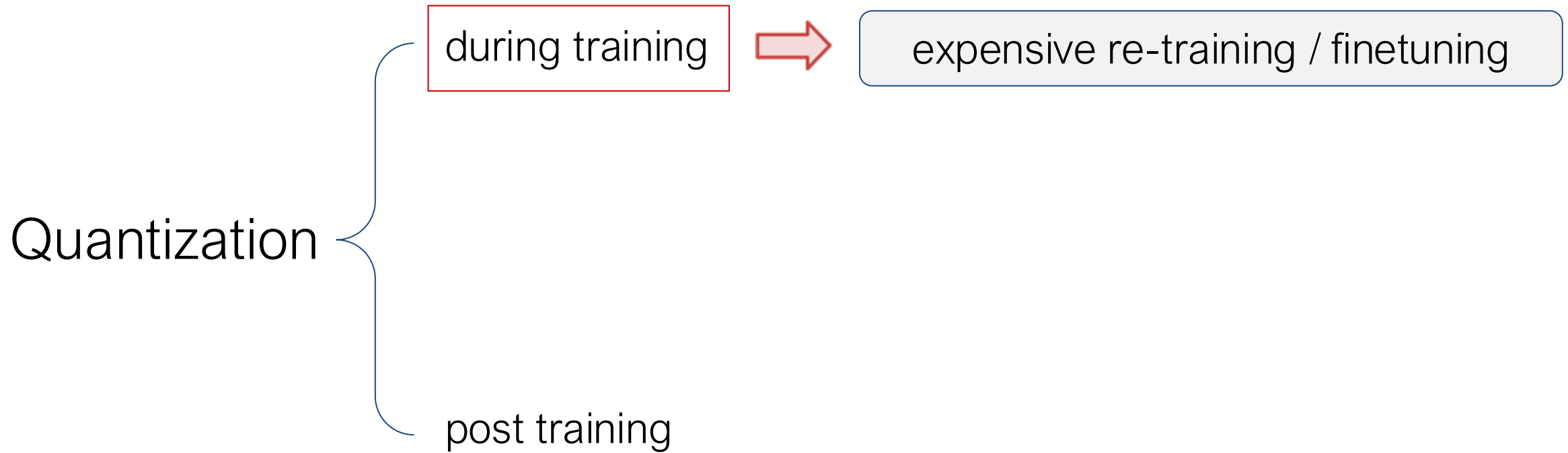
<https://colab.research.google.com/drive/1DPr4mUQ92Cc-xf4GgAaB6dFcFnWlvqYi?usp=sharing>

Today's Topic

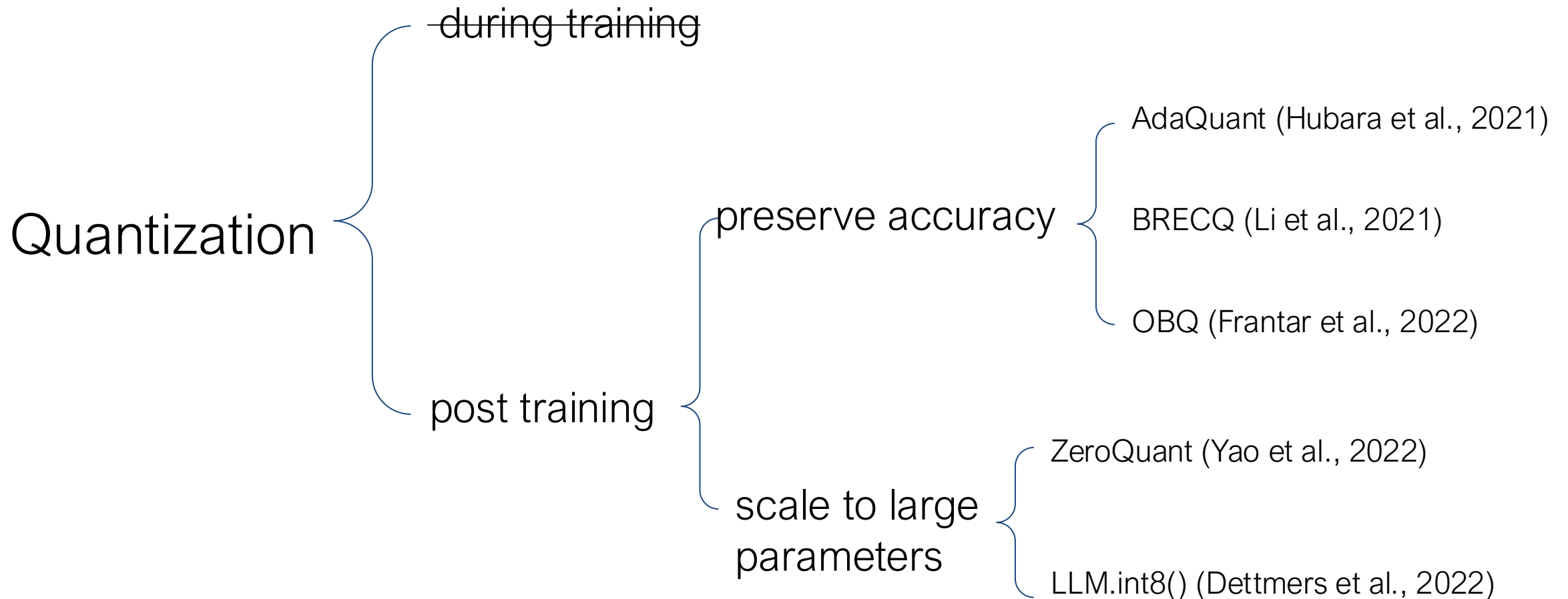
- Low precision numbers in computer

→ • Basic Quantization Methods

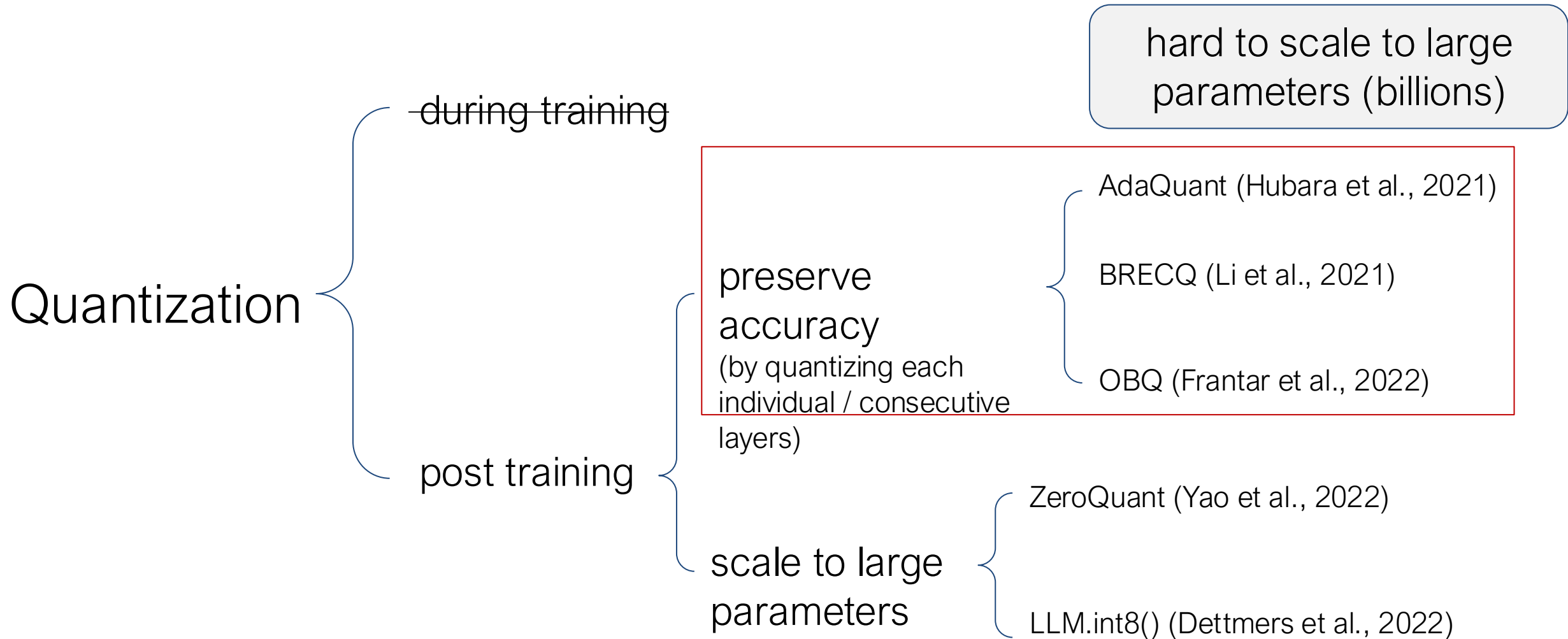
Model Quantization Approaches



Model Quantization Approaches



Model Quantization Approaches



Basic Approach: Layer-Wise Quantization

- Find the quantized matrix \hat{W} that minimizes the linear layer's output.

- Performs quantization layer-by-layer linear projection

$$\operatorname{argmin}_{\hat{W}} \|WX - \hat{W}X\|_2^2$$

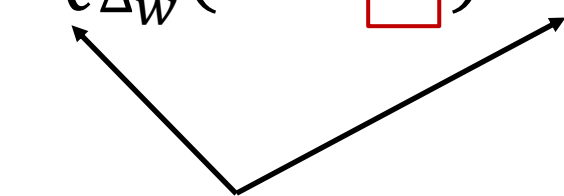
W : linear projection weights (e.g. in FFN and attention)

X : layer input

- Limitation: could still lead to accumulation of quantization error

AdaQuant

- minimize the error between the quantized / full-precision layer outputs for each layer
- adding continuous V to W and quantize

$$(\widehat{\Delta}_W, \widehat{\Delta}_x, \widehat{V}) = \underset{\Delta_W, \Delta_x, V}{\operatorname{argmin}} \|WX - Q_{\Delta_W}(W + V) \cdot Q_{\Delta_x}(X)\|^2$$


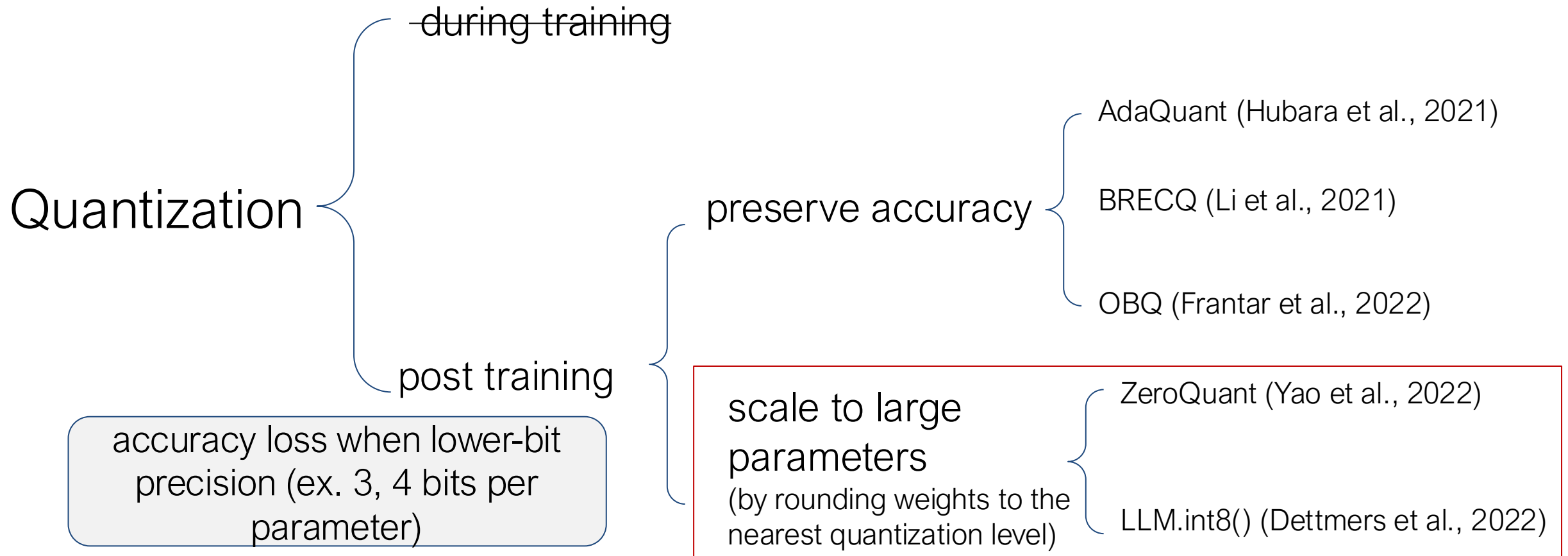
quantized results

the quantization step size Δ_W and Δ_x can be set or learned.

Is Quantization Accurate?

MobileNet-v2	Float Model	Direct Quantized Model	Quantized Model with ADAQUANT
Model Size	8.4 MB	2.3 MB	2.4 MB
P rec@1	65.4 %	1.7 %	52.3 %
P rec@5	85.7 %	5.6 %	75.7 %

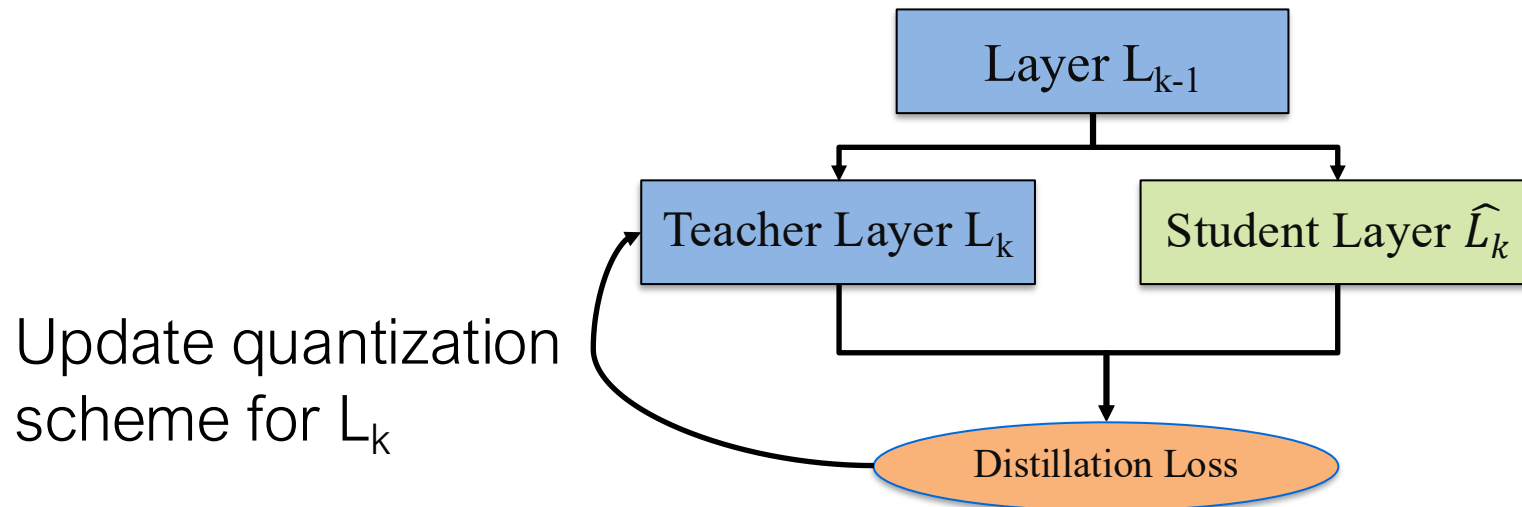
Why is Quantizing LLMs Difficult?



ZeroQuant

- Layer-by-layer knowledge distillation
 - Use the original model as Teacher
 - the quantized model is student

$$\mathcal{L}_{LKD,k} = \left| L_k \cdot L_{k-1} \cdot L_{k-2} \cdot \dots \cdot L_1(X) - \hat{L}_k \cdot L_{k-1} \cdot L_{k-2} \cdot \dots \cdot L_1(X) \right|^2$$



ZeroQuant

- Quantization-Optimized Transformer Kernels (fusion)

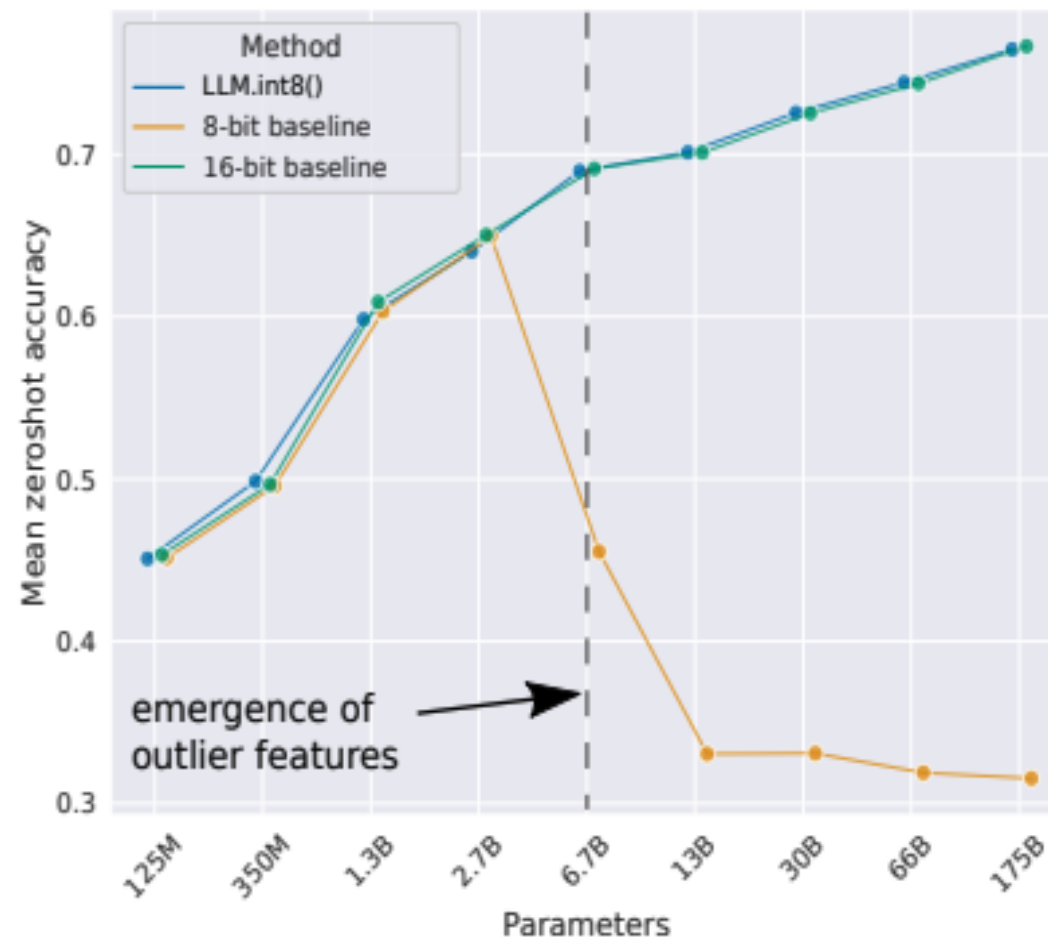


Figure 2: The illustration of normal (left) and our fused (right) INT8 GeMM.

- The scalability is verified up to 20B models (GPT-NeoX_{20B})
- At 1.3B scale, computation time is ~3 hours
 - but slower than GPTQ (x100 larger in ~4 hours)
- integrated in Deepspeed

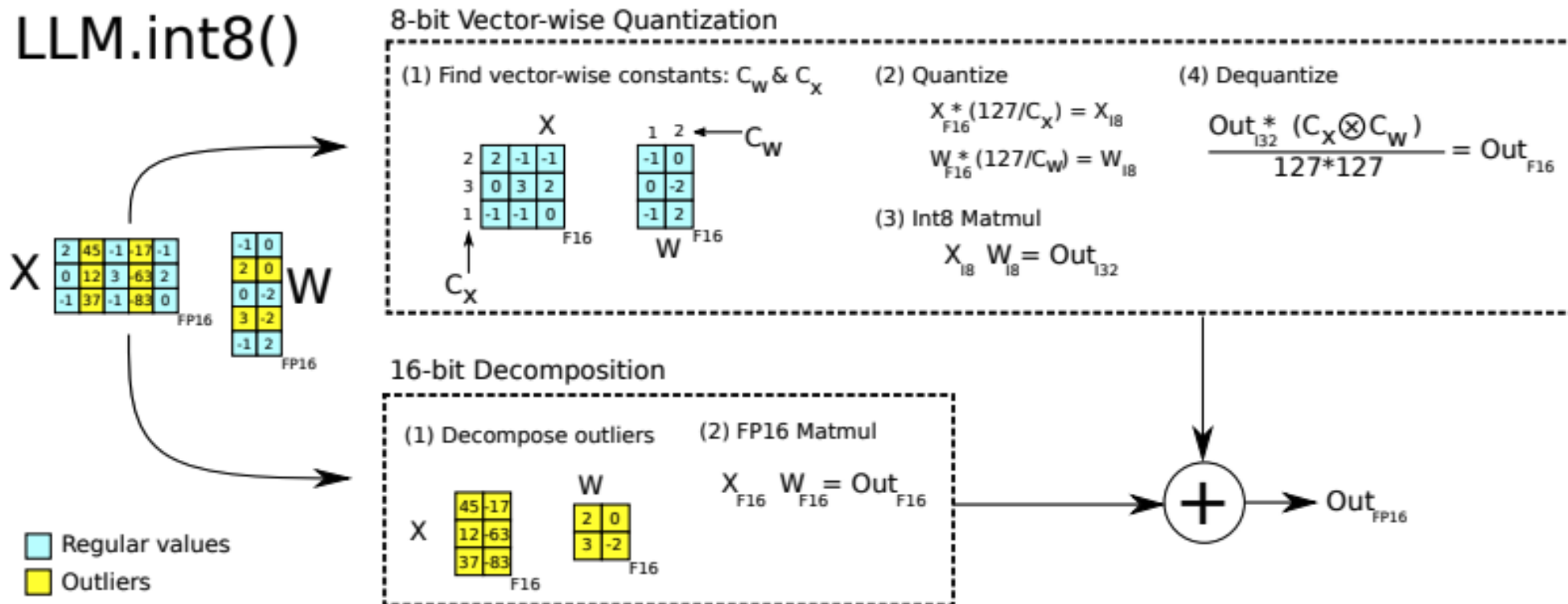
LLM.int8()

- Using 8-bit quantization for matrix multiplications
- But, extreme outliers in features (activation values)
 - need for wider numerical ranges
 - Quantize all parameters without distinguishing them separately can result in accuracy degradation



LLM.int8()

- Keep outliers in higher precision (FP16) while quantizing the rest (8bit)
- Outliers: large magnitude (≥ 6.0), affects $\geq 25\%$ layers, and affects $\geq 6\%$ sequence dimensions



Summary

- low-bit number representation in computer
 - BF16: 16-bit half precision floating point numbers, better for ML
 - int8
- Direct quantization
 - absmax: linearly scale according to max abs value
 - zero-point: finding zero-point and scale
- Layer-wise quantization approaches
 - AdaQuant
 - KD: ZeroQuant
 - LLM.int8()

Next

- Scaling Quantization for large models: GPTQ