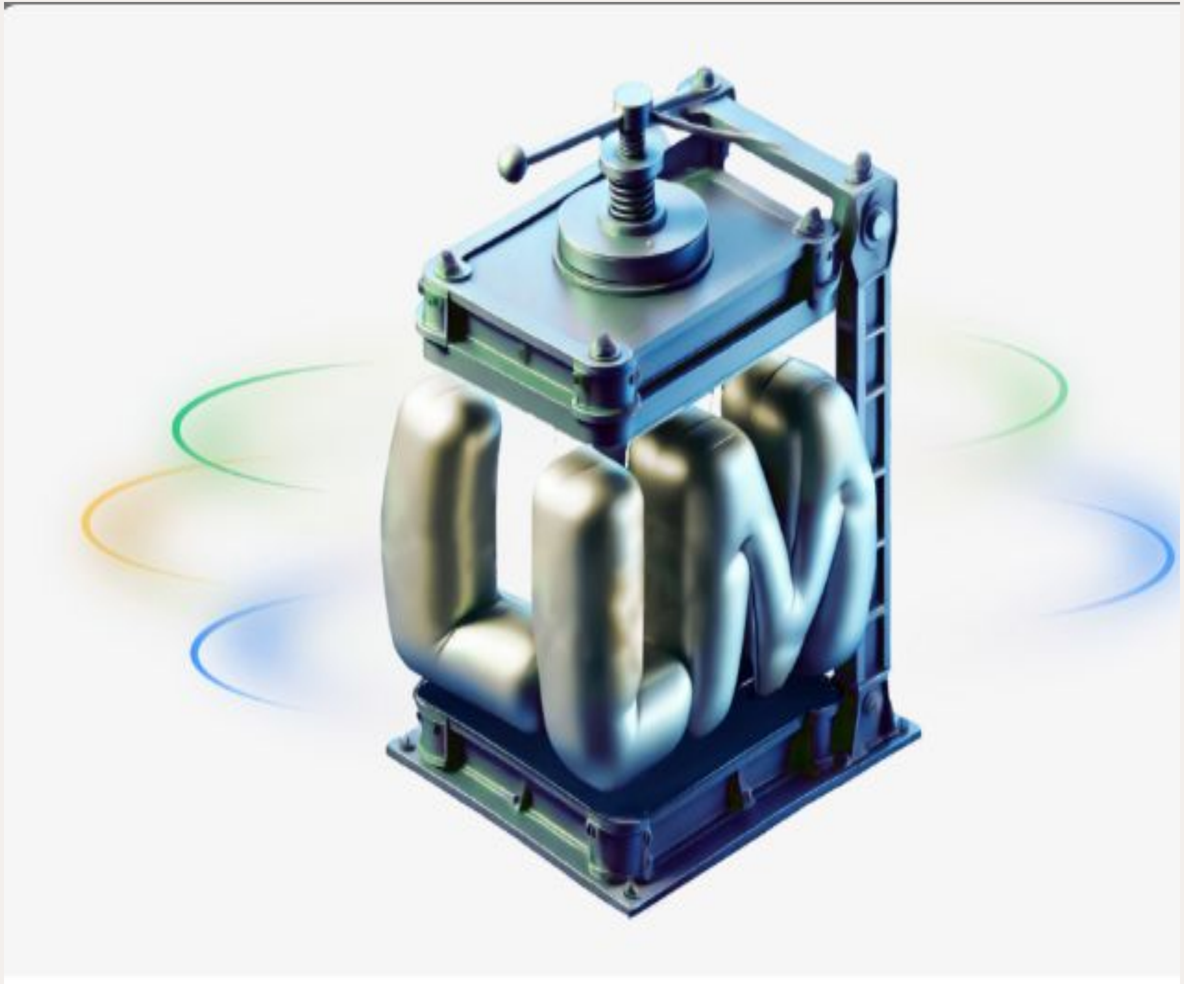
BTP - GROUP_36

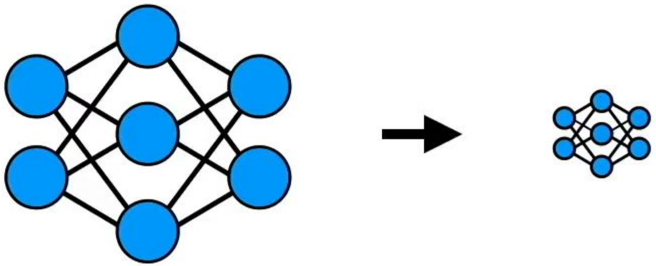
-Jai Anurag
-Nishant yadav

Topic: Efficient Storage of Large Language Models on Edge Devices



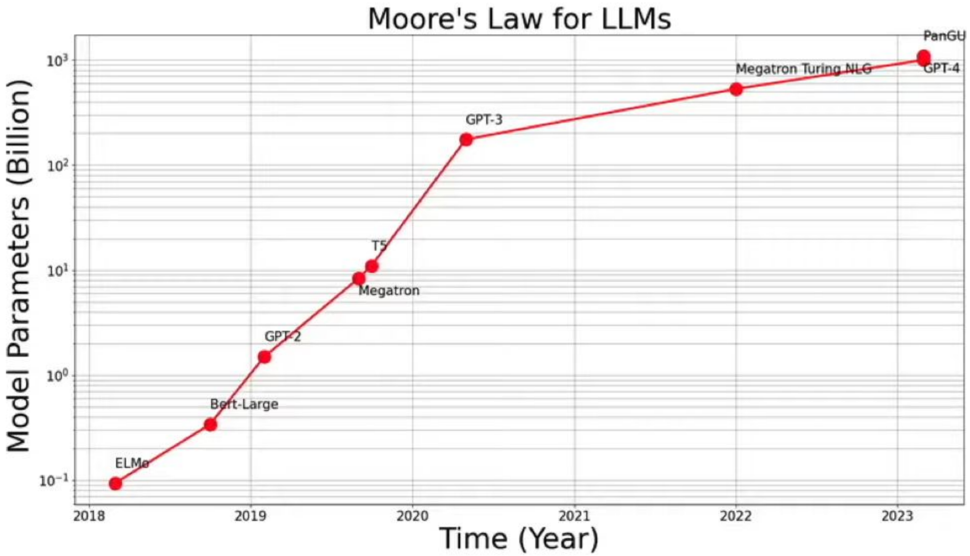
Compressing LLMs

Make models 10X smaller without sacrificing performance



“Bigger is Better”

More Data + More Parameters + More Compute = Better Models



The Problem

Bigger models mean higher costs

100B Params $\Longrightarrow$ 200GB storage (FP16)

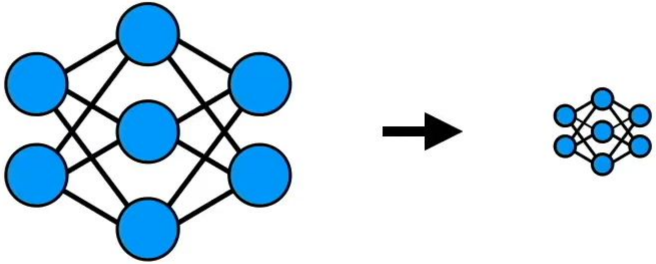
High Compute

High Cost

High CO_2

Model Compression

Reduce ML model size without sacrificing performance



Less Compute

Less Cost

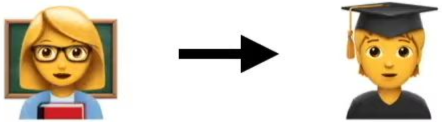
Less CO_2

3 Ways to Compress LLMs

1) Quantization

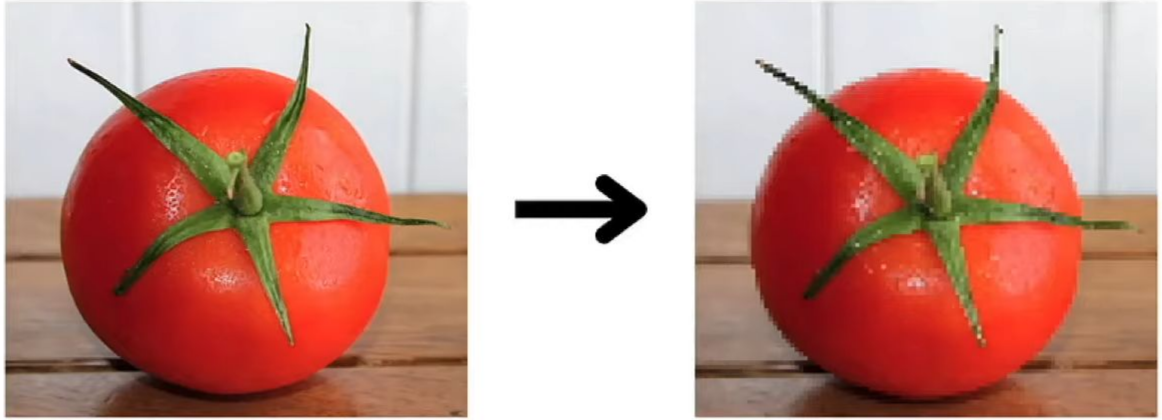
2) Pruning

3) Knowledge Distillation



1) Quantization

Lowering the precision of model parameters



FP32

01000000111000000000000000000000

INT8

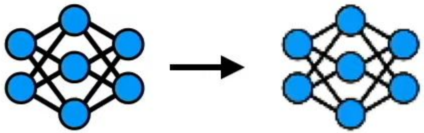
00000111

1) Quantization

Lowering the precision of model parameters

Post-training Quantization

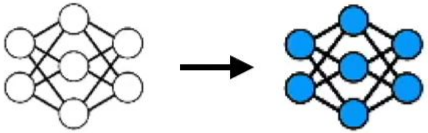
Train then Quantize



FP32 → 8-bit, 4-bit

Quantization-Aware Training

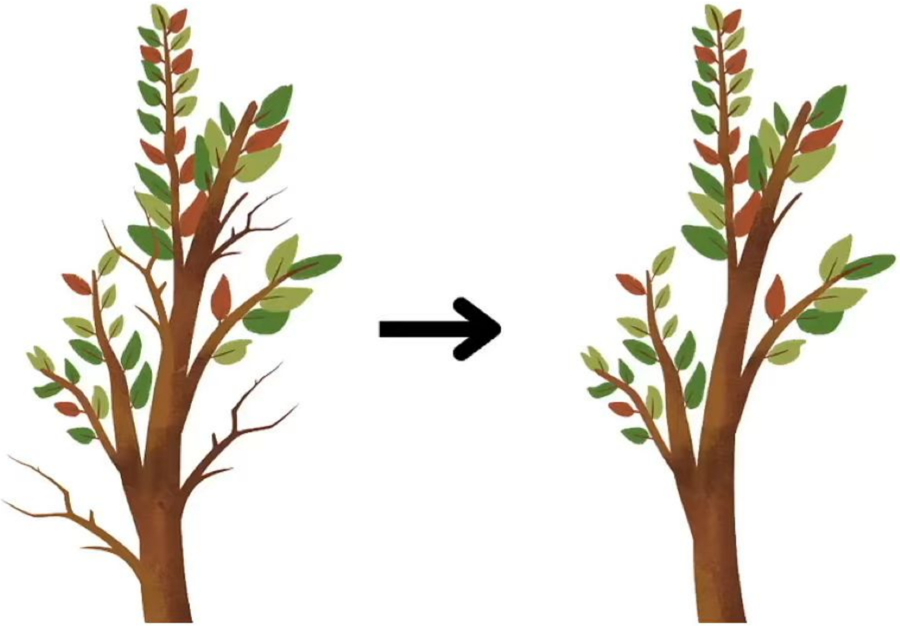
Quantize then train



4-bit and lower

2) Pruning

Removing unnecessary components from a model



100B Params

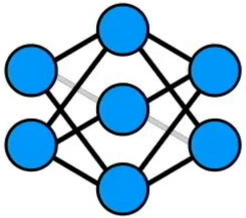
60B Params

2) Pruning

Removing unnecessary components from a model

Unstructured

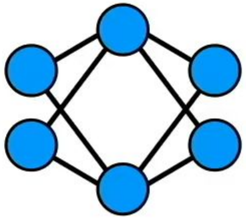
Remove individual weights



Greater reduction, but
requires specialized hardware

Structured

Remove entire structures
(e.g. attention heads, neurons, layers)



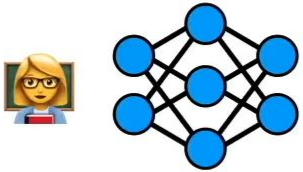
Less reduction, but parameters
can be removed entirely

3) Knowledge Distillation

Transfer knowledge from (larger) teacher model to (smaller) student model

Soft Targets

Train student using logits



Positive: 0.85

Negative: -0.85

Note: for NLG model will output 50k logits (one for each token)

Prediction Target

$$\begin{bmatrix}0.75\\-0.75\end{bmatrix} \longleftrightarrow \begin{bmatrix}0.85\\-0.85\end{bmatrix}$$

Synthetic Data

Train student using text

ChatGPT

Input:
<|im_start|>system You are a helpful assistant<|im_end|>
<|im_start|>user
What is the capital of France?<|im_end|>
<|im_start|>assistant

Output:
Paris<|im_end|>

Alpaca

Train on
(Input, Output)
pairs

QLoRA (Quantized Low-Rank Adaptation)

LLM fine-tuning made accessible

Fine-tuning (recap)

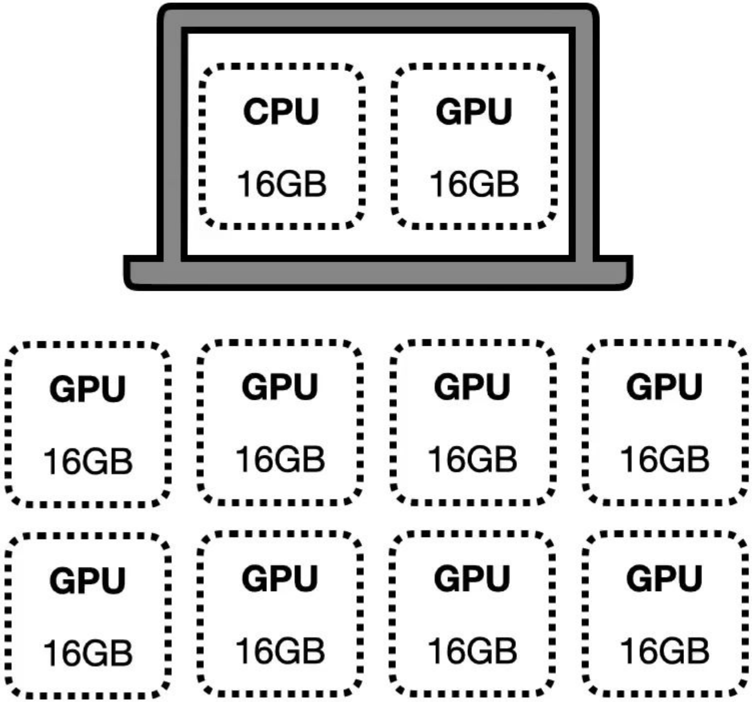
Tweaking an existing model for a particular use case.

GPT-3

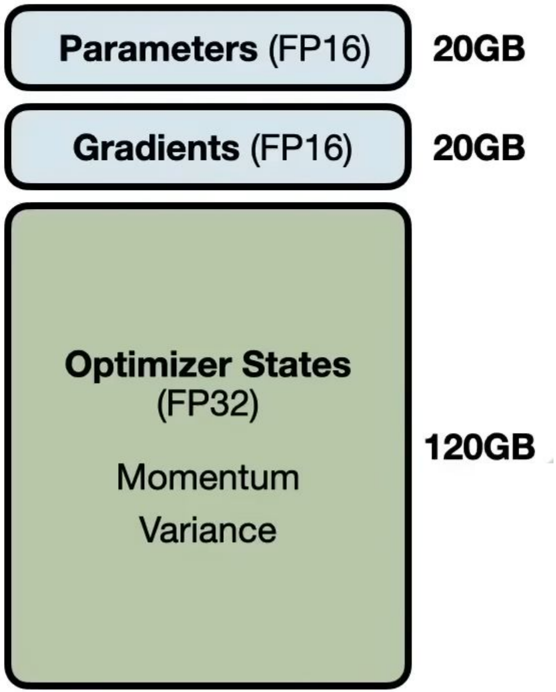
ChatGPT

The Problem

LLMs are (computationally) expensive

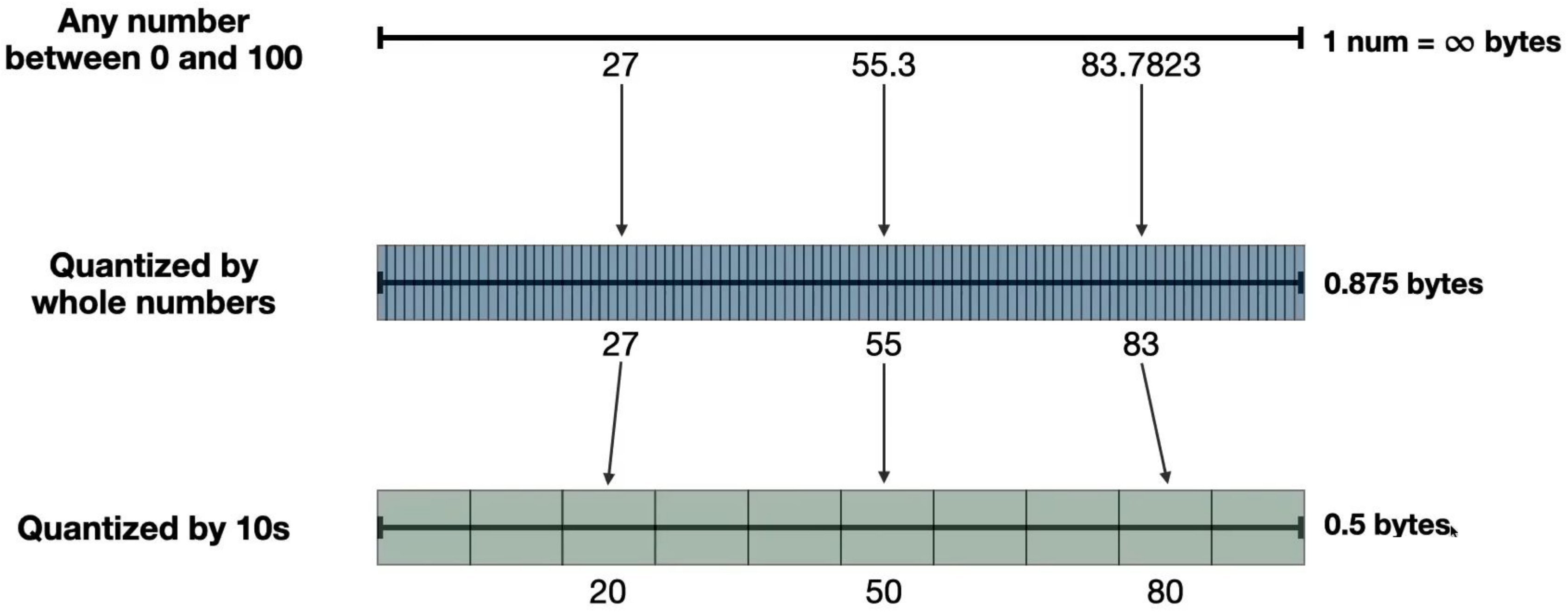


10B Parameter Model = 160GB!



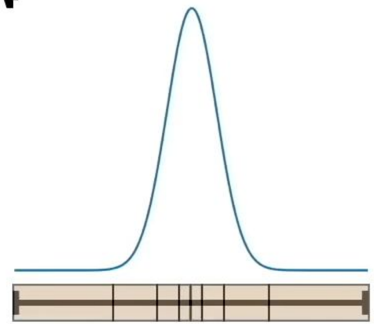
What is Quantization?

Quantization = splitting range into buckets



4 Ingredients of QLoRA

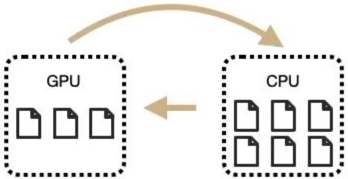
1. 4-bit NormalFloat



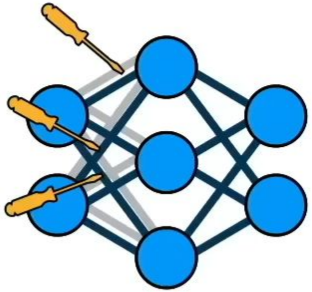
2. Double Quantization

quant(quant())

3. Paged Optimizers



4. LoRA



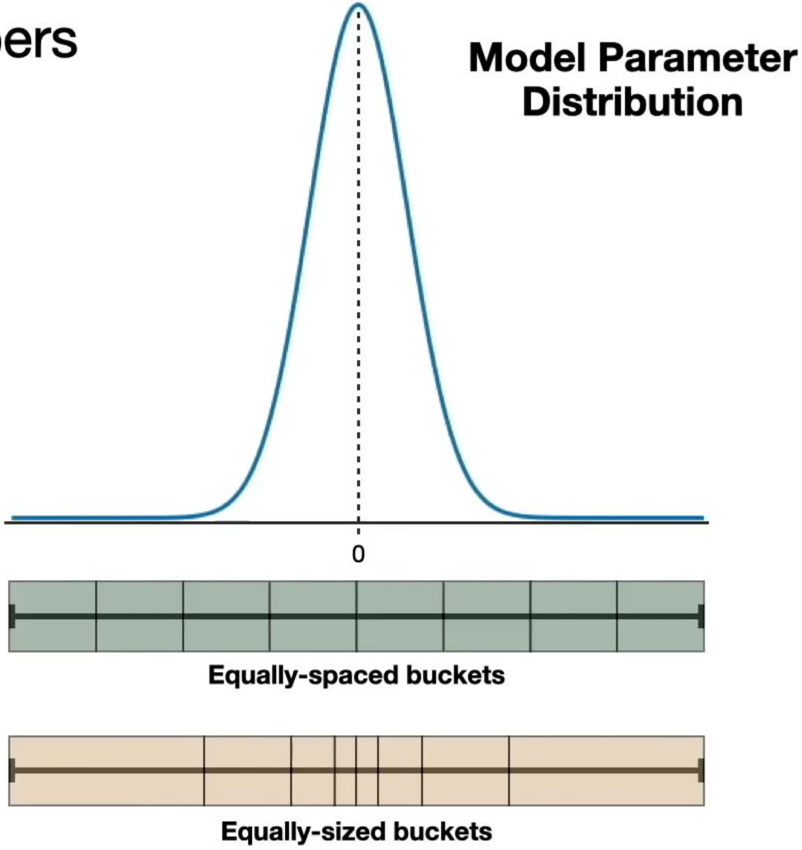
Ingredient 1: 4-bit NormalFloat

A better way to bucket numbers

4-bit e.g. 0101

$\Longrightarrow 2^4 = 16$ unique combinations

$\Longrightarrow 16$ buckets for quantizations



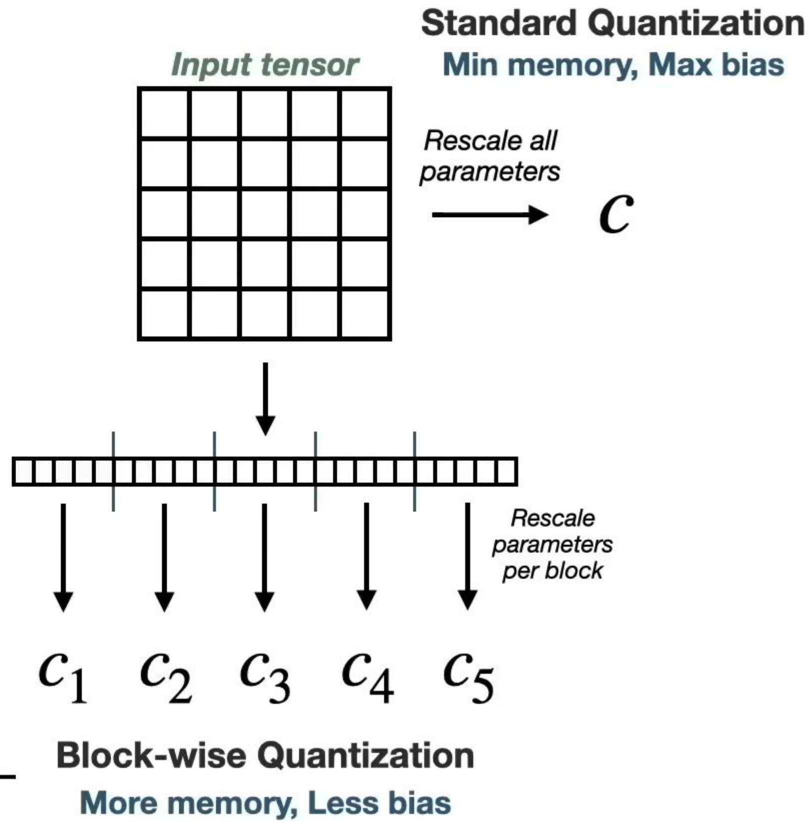
Ingredient 2: Double Quantization

Quantizing the Quantization Constants

$$X^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(X^{\text{FP32}})} X^{\text{FP32}}\right)$$

$$= \text{round}\left(c^{\text{FP32}} \cdot X^{\text{FP32}}\right)$$

Takes up precious memory

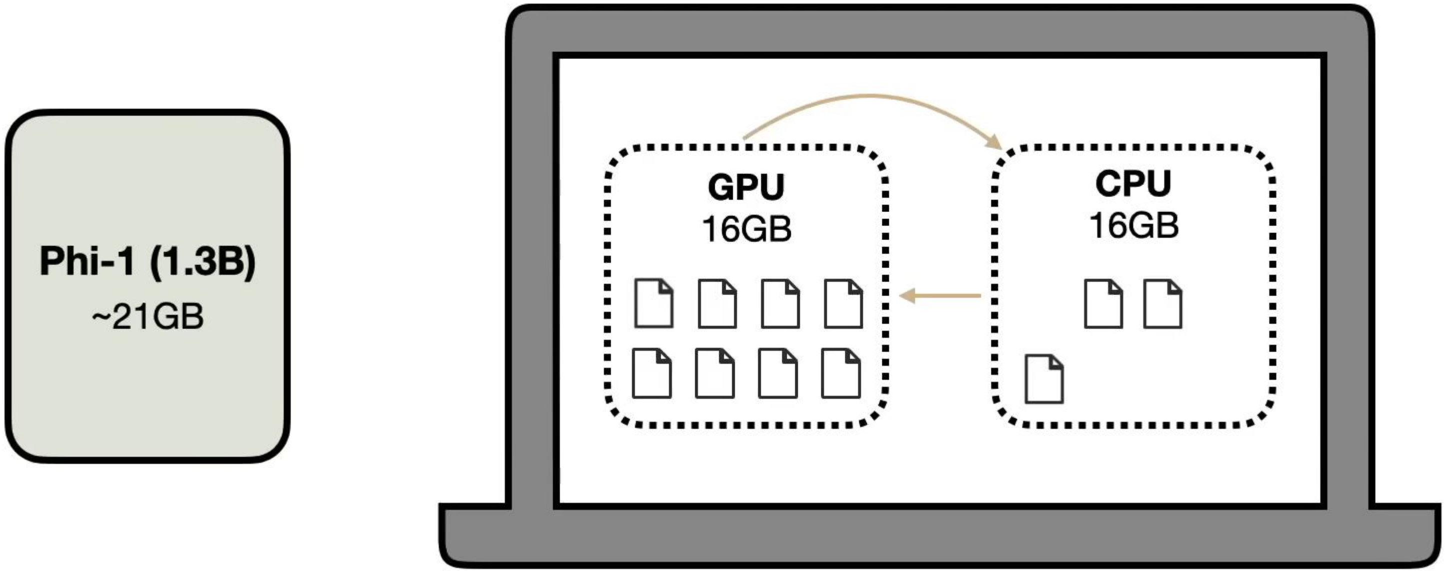


Double Quantization

$$C^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(C^{\text{FP32}})} C^{\text{FP32}}\right)$$

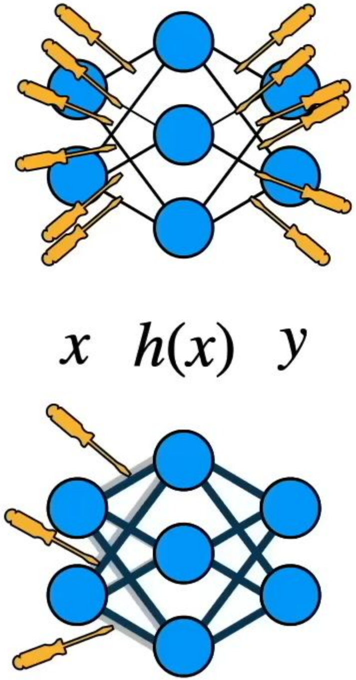
Ingredient 3: Paged Optimizer

Looping in your CPU

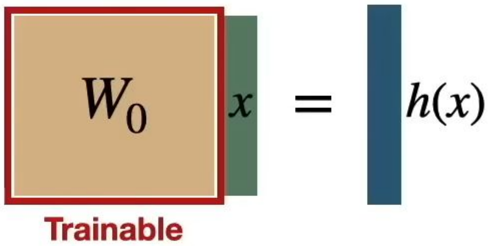


Ingredient 4: LoRA

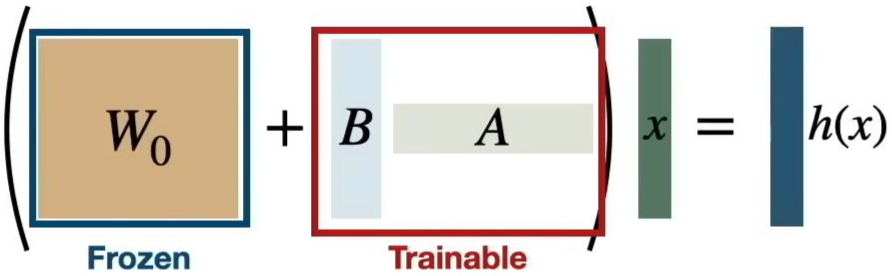
Fine-tunes model by adding **small set** of trainable parameters



Full Fine-tuning: $h(x) = W_0 x$

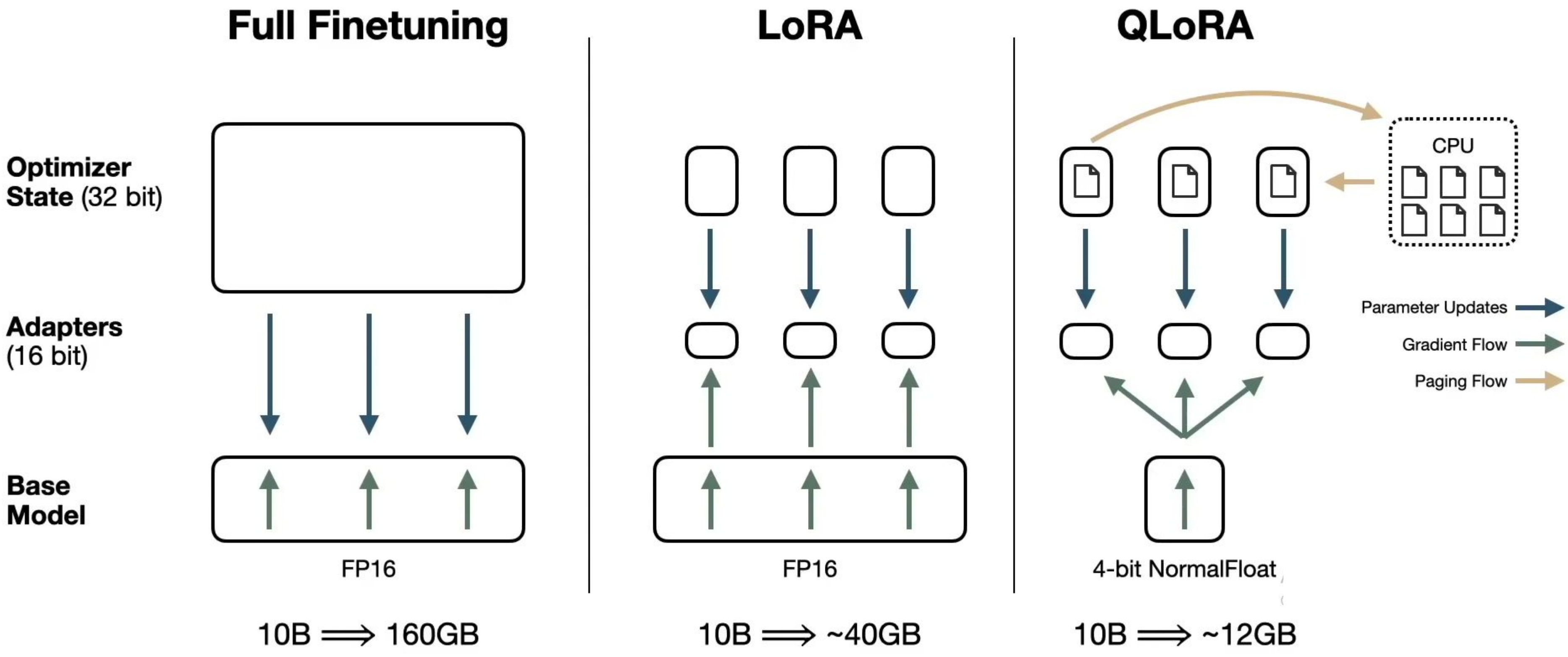


LoRA: $h(x) = W_0 x + \Delta W x = W_0 x + BAx$



100-1000X savings!

Bringing it all together



Storage Strategies for LLMs on Edge Devices

Optimizing Model Storage for Resource-Constrained Environments

"How to fit GB-sized LLMs into MB-sized storage?"

Storage Solutions for Edge Devices

Hardware-Specific Storage Strategies

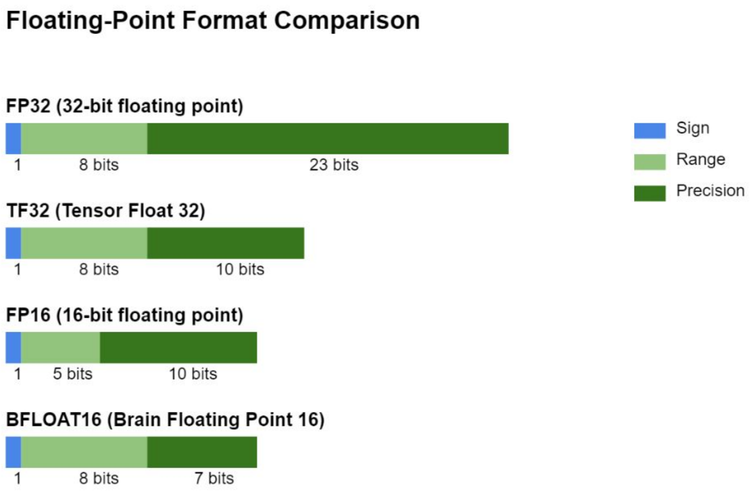
1. **Flash Memory / eMMC / microSD** (RPi, Jetson Nano)
   - Use **compressed formats** (GGUF, ONNX, TFLite).
   - Example: *Llama 3-8B (quantized) on 128GB SD card*.
2. **NOR/NAND Flash** (MCUs: ESP32, STM32)
   - TinyML: **<10MB models** (e.g., Mistral-Tiny).
   - Example: *Keyword detection on ESP32*.
3. **eDRAM/SRAM** (NPUs: EdgeTPU, Coral)
   - **Model tiling**: Split weights between SRAM/Flash.
   - Example: *Pruned LLM on Coral Dev Board*

Efficient LLM Storage Techniques, Shrinking LLMs for Edge Deployment

1. **Quantization**
   - FP32 → INT8 (4x smaller) or INT4 (8x smaller).
   - *Tools: GPTQ, AWQ, TensorRT*.

2. **Weight Pruning**
   - Remove redundant neurons (e.g., 8B → 4B model).

3. **Adapter Layers (LoRA/PEFT)**
   - Store only fine-tuned layers (MBs instead of GBs).



Bullet Points:

- Match **storage type** to device (eMMC for SBCs, Flash for MCUs).
- Always **quantize & prune** models before deployment.
- Use **adapters (LoRA)** for personalized fine-tuning.
- Offload to **external media** (SSD > USB > SD Card) for large models.

Tools/Frameworks:

- Quantization: *TensorRT, GPTQ*
- Compression: *GGUF, ONNX*
- Adapters: *LoRA, PEFT*

“Our Future Goals”

Real-Time Edge AI: Sensor-Driven LLMs
Connecting Cameras, Microphones & More to Optimized LLMs

Goal: *"Process multimodal inputs locally with low latency, no cloud dependency."*

Use Cases & Technical Pipeline

Applications & Data Flow

1. **Use Cases** :
    - **Smart Surveillance**: Anomaly detection (intruders, fires).
    - **AR Assistants**: Scene description for visually impaired.
    - **Multimodal Chatbots**: Smart doorbells with voice+image understanding.
2. **Data Pipeline**:
    - **Step 1**: Sensor → Preprocessing (frame → embeddings).
    - **Step 2**: Adaptive sampling (skip frames if resource-constrained).
    - **Step 3**: Streaming inference (token-by-token generation).

Optimization Strategies

Making It Work on Edge Devices

1. **Model Compression**
   - Quantization (FP32 → INT4: 8x smaller).
   - Pruning (remove 50% weights with <1% accuracy drop).

2. **Edge Runtimes**
   - TensorRT (NVIDIA), OpenVINO (Intel), TVM (ARM).

3. **On-Device Adaptation**
   - LoRA fine-tuning (store only 2MB adapters, not full model).
   - Federated learning (collaborative edge training).

4. **Hybrid Edge-Cloud**
   - Offload complex tasks to cloud; edge handles real-time.

References:

1) https://arxiv.org/pdf/2305.14314
2) https://arxiv.org/pdf/1503.02531
3) https://arxiv.org/pdf/2106.09685
4) https://github.com/artidoro/qlora

- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer