LLM Efficiency

Tomasz Limisiewicz

18 April 2024

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Lesson plan

1. 💬 **Assignments** (20 min)
2. 🍾 **Discussion + Bottlenecks** in Transformers (30 min)
3. 🏃 **Efficiency** algorithms (30 min)
4. 🌐 **LLMs beyond English** (30 min next week)

Discussion

What is the main problem when training a LLM?

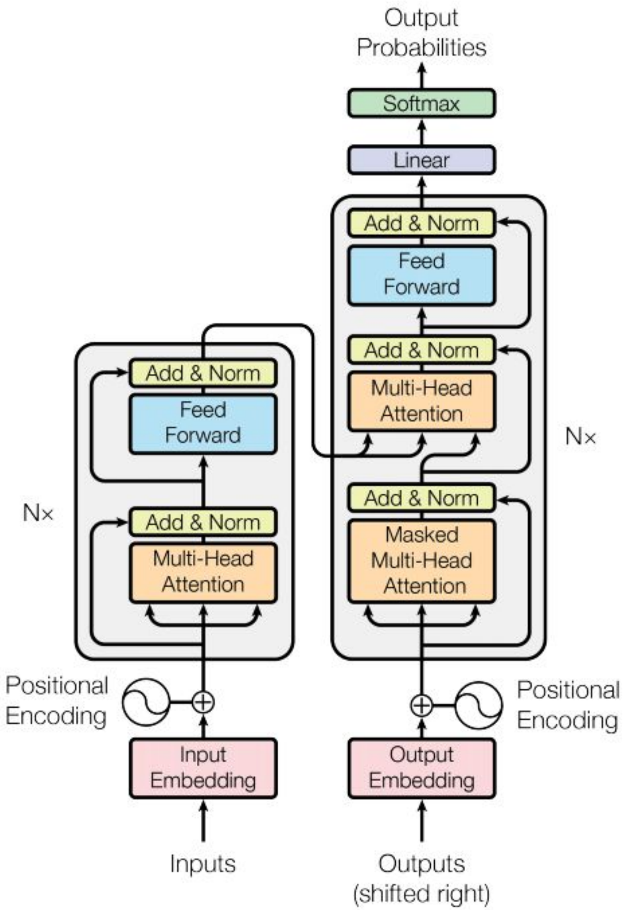
computation speed? **memory**? **disk space**?

Transformer Bottlenecks

Speed

Complexities against:
inner dimension (d) and
sequence length (n), vocabulary (v).

- Feed forward: $O(n \cdot d^2)$
- Linear Softmax: $O(n \cdot v \cdot d)$
- Attention: $O(n^2 \cdot d)$

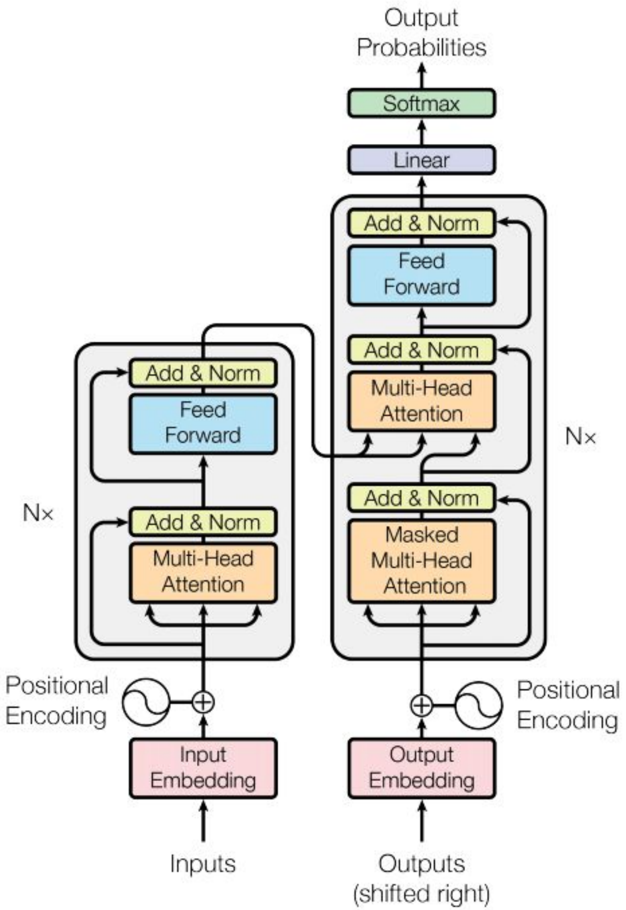


Speed

Complexities against:
inner dimension (d) and
sequence length (n), vocabulary (v).

- Feed forward: $O(n \cdot d^2)$
- Linear Softmax: $O(n \cdot v \cdot d)$
- Attention: $O(n^2 \cdot d)$

Complexity of transformer:
$O(n^2 \cdot d + n \cdot d^2)$



Disk Space

Size of the model:

Usually 2 bytes per parameter (16 bit)

10B param model ➡️ 20GB

Disk Space

Size of the model:

Usually 2 bytes per parameter (16 bit)

10B param model ➡️ 20GB

Size of dataset:

Pre-training: ~10T tokens ➡️ 50TB

Fine-tuning and inference: up to 100s GB

Memory

Inference:

Usually 2 bytes per parameter (16 bit)

10B param model ➡️ 20GB (+ some for inference batch)

Memory

Inference:

Usually 2 bytes per parameter (16 bit)

10B param model ➡️ 20GB (+ some for inference batch)

Training or Fine-tuning:

2 bytes per parameter
2 bytes per gradient

Memory

Inference:

Usually 2 bytes per parameter (16 bit)

10B param model ➡️ 20GB (+ some for inference batch)

Training or Fine-tuning:

2 bytes per parameter
2 bytes per gradient
12 bytes per optimizer weight (Adam)

10B param model ➡️ ❓

Memory

Inference:

Usually 2 bytes per parameter (16 bit)

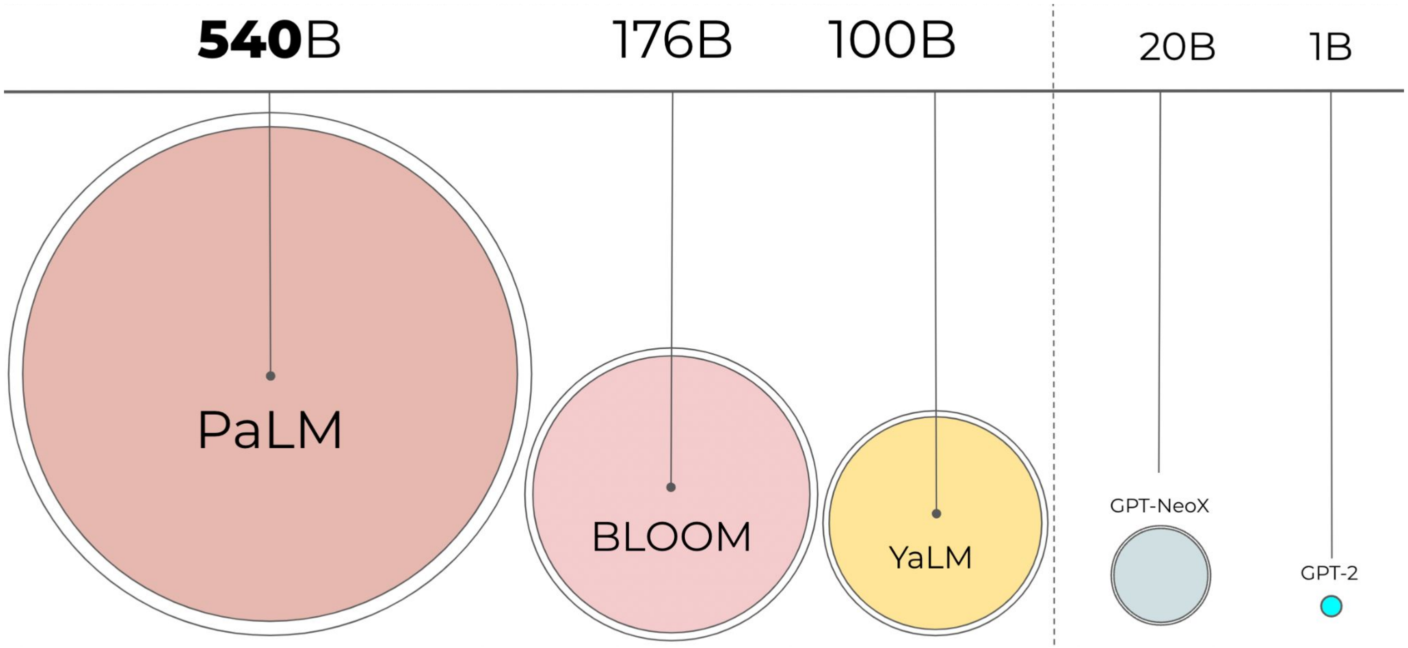
10B param model ➡️ 20GB (+ some for inference batch)

Training or Fine-tuning:

2 bytes per parameter
2 bytes per gradient
12 bytes per optimizer weight (Adam)

10B param model ➡️ **160GB !**

Memory



10B param model ➡️ **160GB !**

Efficiency Algorithms

Problem: Constrained Memory in Training

What can we do? Any ideas?

Problem: Constrained Memory in Training

What can we do? Any ideas?

💡 Get few million dollars to buy a brand new GPU cluster!

Problem: Constrained Memory in Training

What can we do? Any ideas?

💡 Get few million dollars to buy a brand new GPU cluster!

💡 Tune only some parameters for specific task

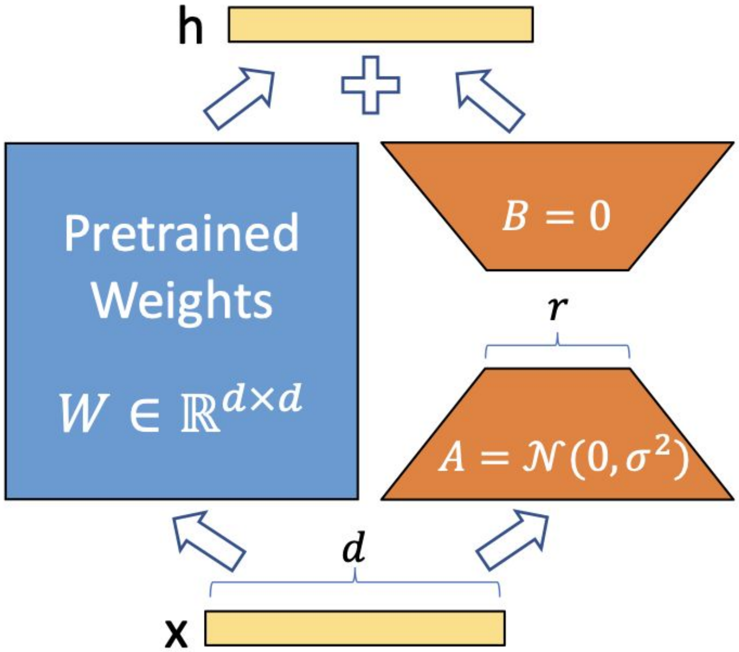
💡 Use smaller parameters

💡 Use smaller models

Parameter Efficiency in Fine Tuning: LoRA

source: https://arxiv.org/pdf/2106.09685

Weight matrices are decomposed



Parameter Efficiency in Fine Tuning: LoRA

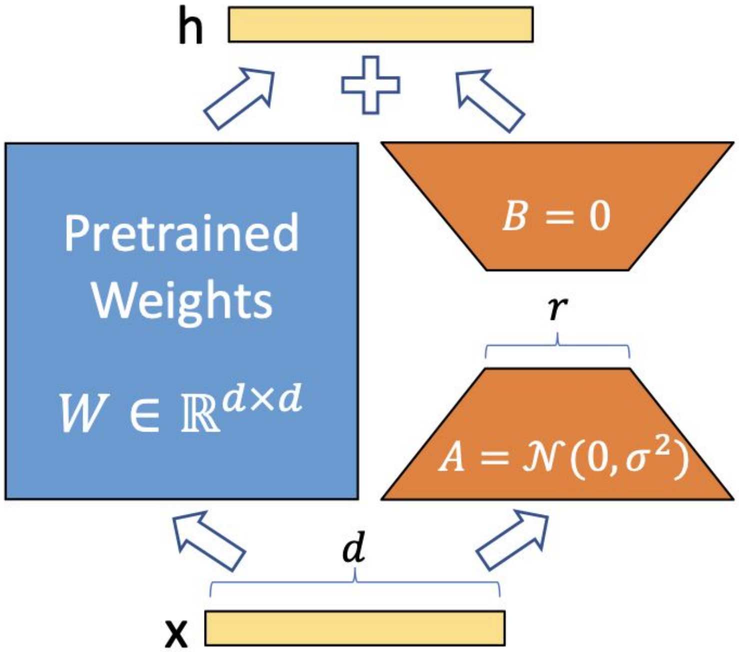
source: https://arxiv.org/pdf/2106.09685

Weight matrices are decomposed

Decrease tunable parameters by 1000 to 10000 times

Gradient computed just for adapted parameters (not whole model)

Originally only attention layers were adapted in LoRA



If you prefer equations:

$$h = W_0x + \Delta W x = W_0 + BAx$$

Where

$$W_0 \in \mathbb{R}^{d\times k}$$

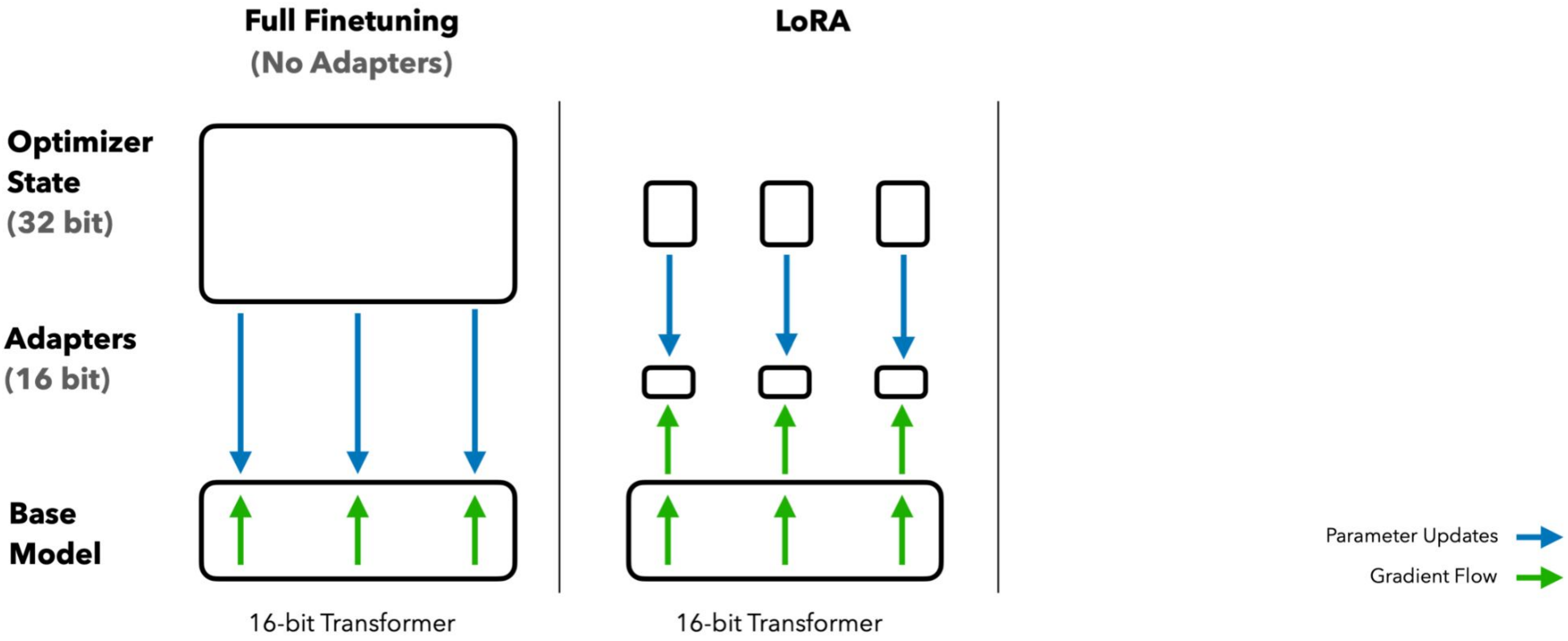
$$r \ll \min(d, k)$$

$$B \in \mathbb{R}^{d\times r} \quad A \in \mathbb{R}^{r\times k}$$

At the beginning of the training B initialized to 0, A initialized randomly.

Fine-Tuning vs. LoRA

source: https://arxiv.org/pdf/2305.14314

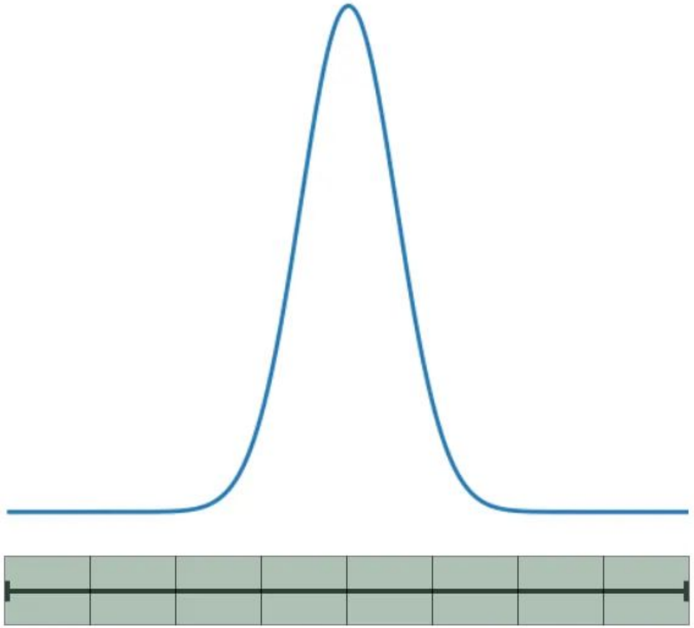


Quantization

The size of parameters may be decreased by quantization

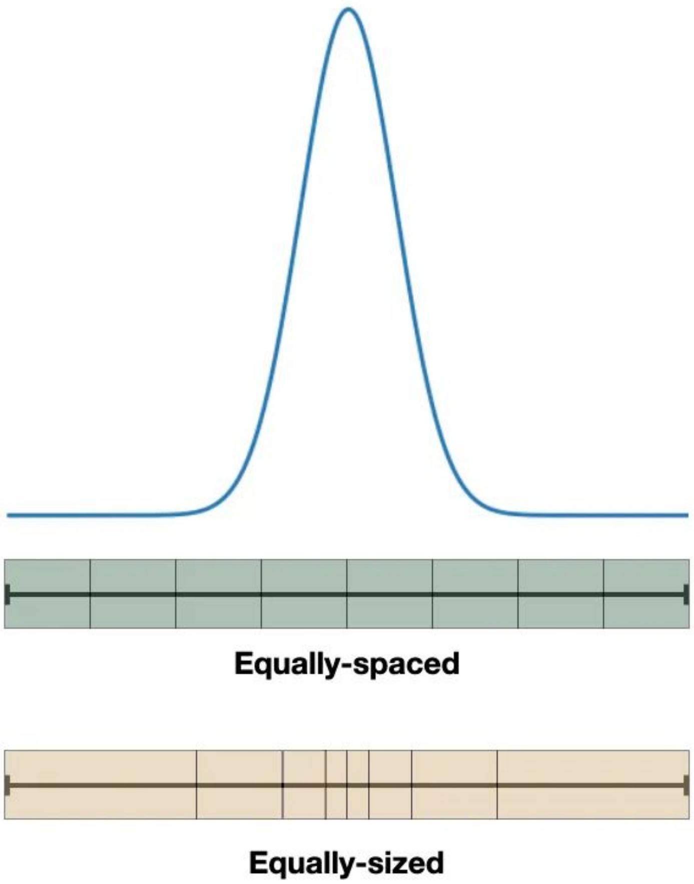
Parameters are assigned into coarse buckets

Important to determine the range of the quantization c



source: https://towardsdatascience.com

4bit NormalBit quantization:
equally-sized buckets based

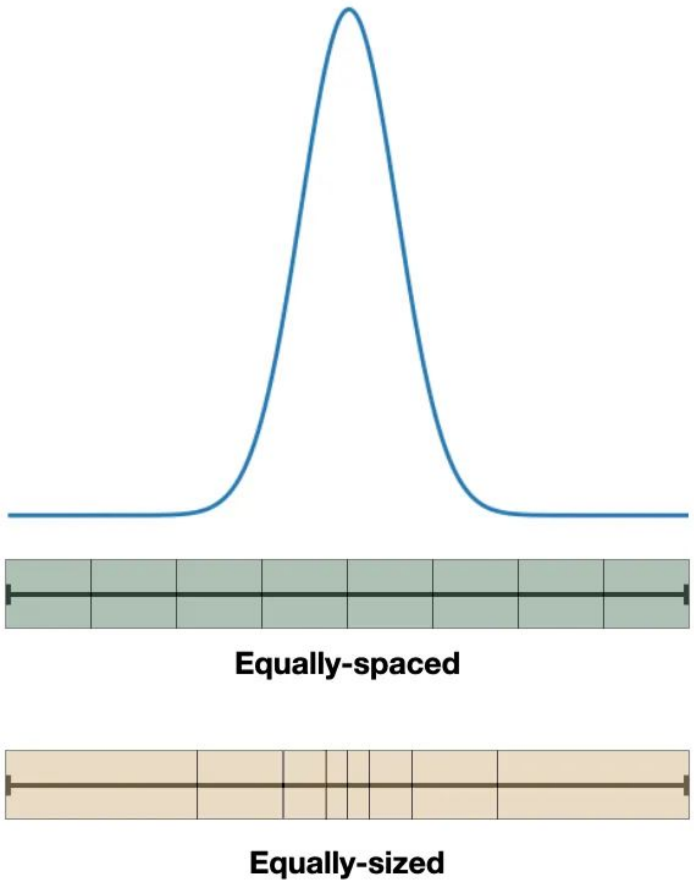


QLoRA

source: https://towardsdatascience.com

4bit NormalBit quantization:
equally-sized buckets based

Double Quantization: quantize both parameters but also their range c

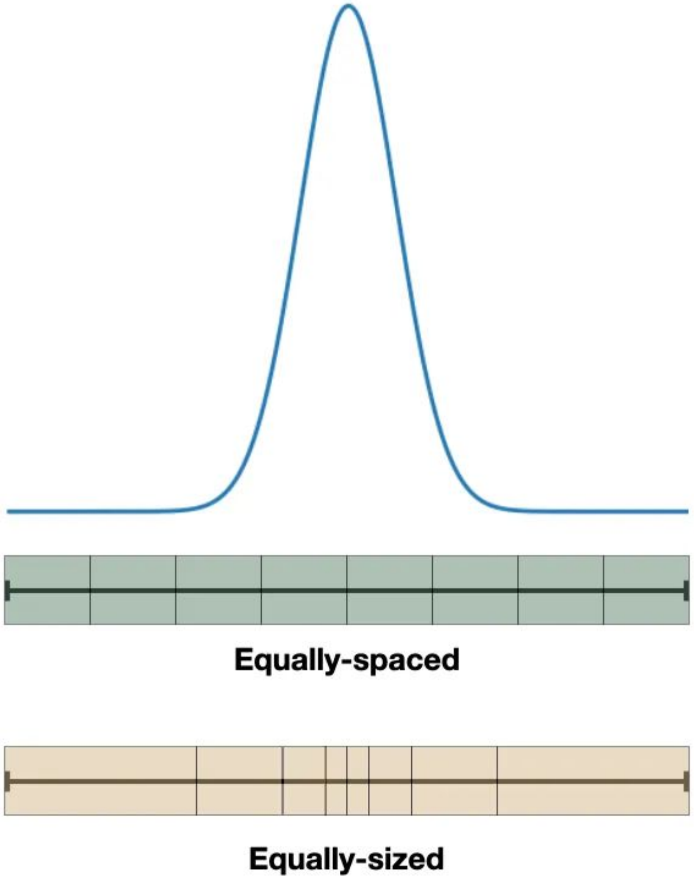


QLoRA

4bit NormalBit quantization: equally-sized buckets based

Double Quantization: quantize both parameters but also their range c

IMPORTANT: LoRA adaptation of all the layers (attention and feed forward)

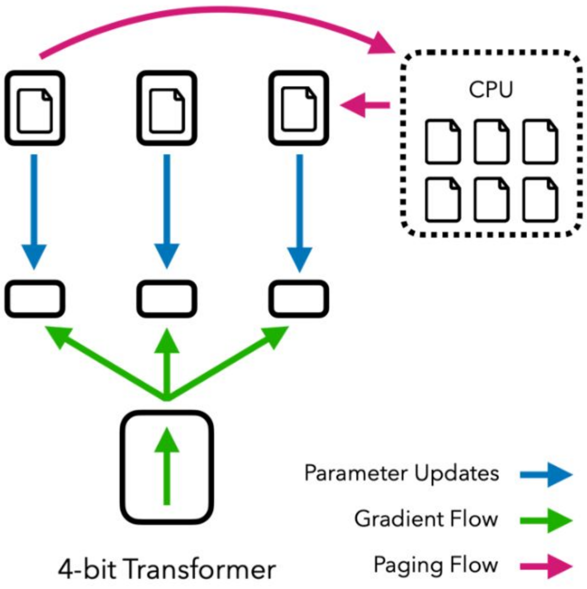


QLoRA: Paged Optimizer

source: https://arxiv.org/pdf/2305.14314

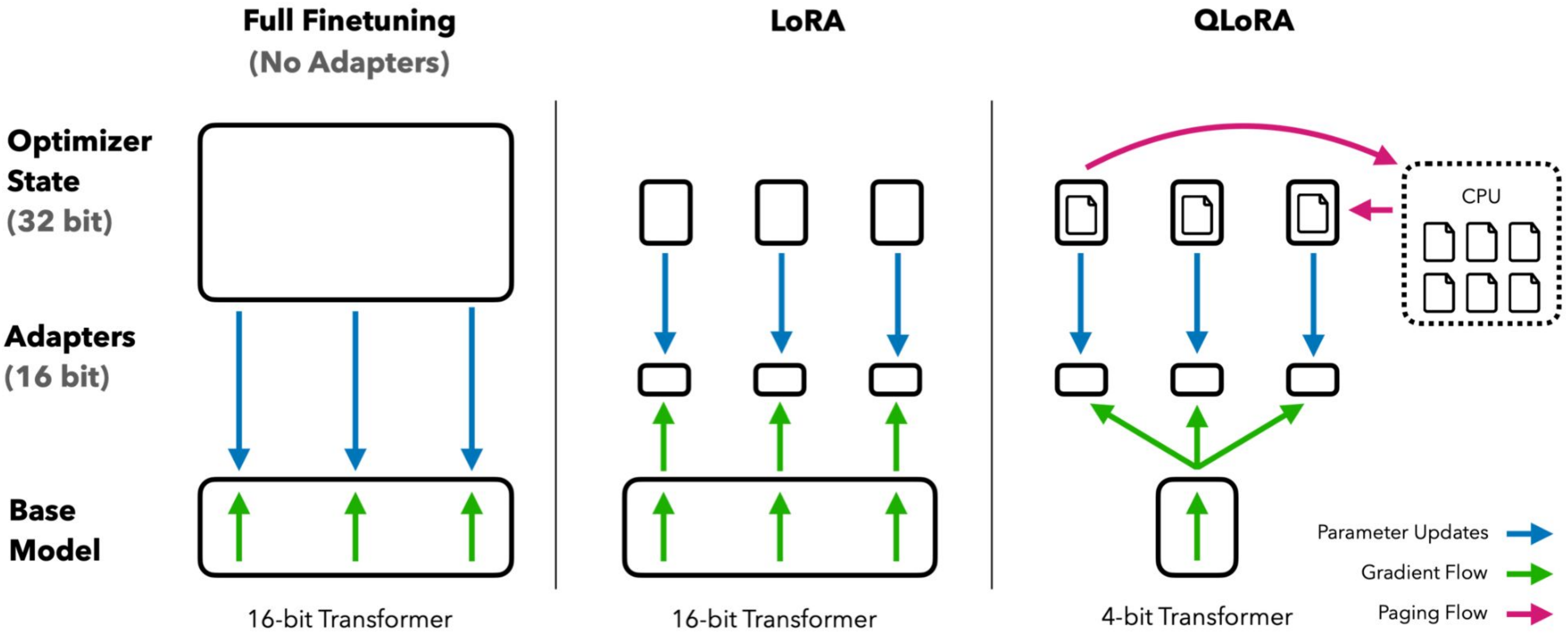
Optimizer weights are transferred between GPU and CPU memory.

It prevents running out of memory when processing long sequences.



Fine-Tuning vs. LoRA vs. QLoRA

source: https://arxiv.org/pdf/2305.14314



Fine-Tuning vs. LoRA vs. QLoRA

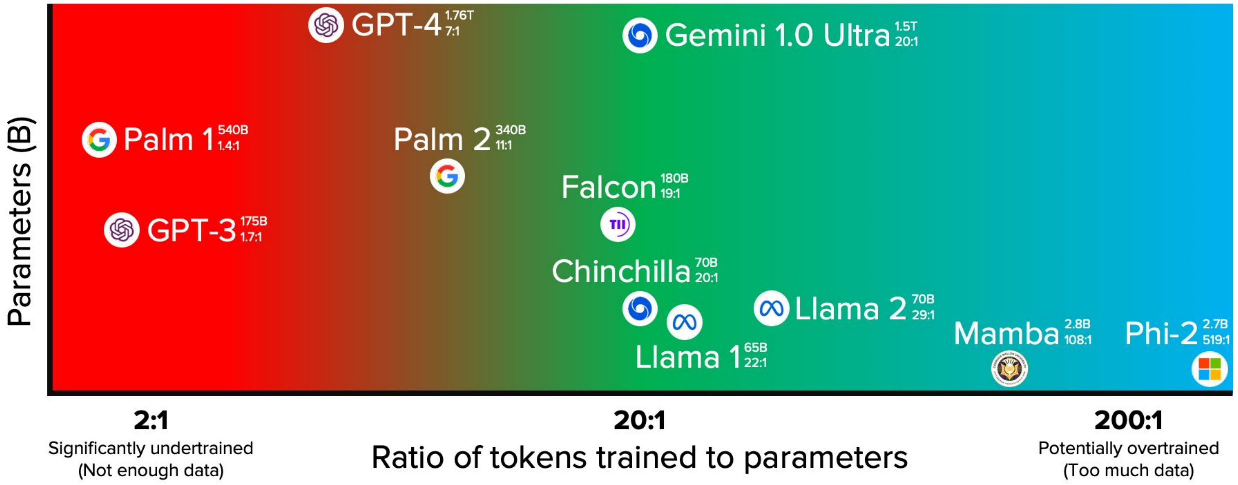
	Fine-Tuning	LoRA	QLoRA
Tunable parameters	100 %	**~0.1%**	**~0.2%**
Model Precision	16 bit	16 bit	**4 bit**
RAM 10B model	160GB	~40GB	**~12GB**
Applicable for	Industrial Supercomputer	Academic Cluster	**Good Personal Setting**
Matches performance	—	**YES**	**YES*** (in full model tuning)

Scale Wisely: Chinchilla Rule

sources: https://arxiv.org/pdf/2203.15556
https://lifearchitect.ai/chinchilla/

Larger models perform better, but what size is enough for me?

It's often better to prioritize data scale over model scale.



Selected highlights only. Mostly to scale. Informed estimates for Palm 2, GPT-4, and Gemini. Alan D. Thompson. November 2022, major update December 2023. https://lifearchitect.ai/

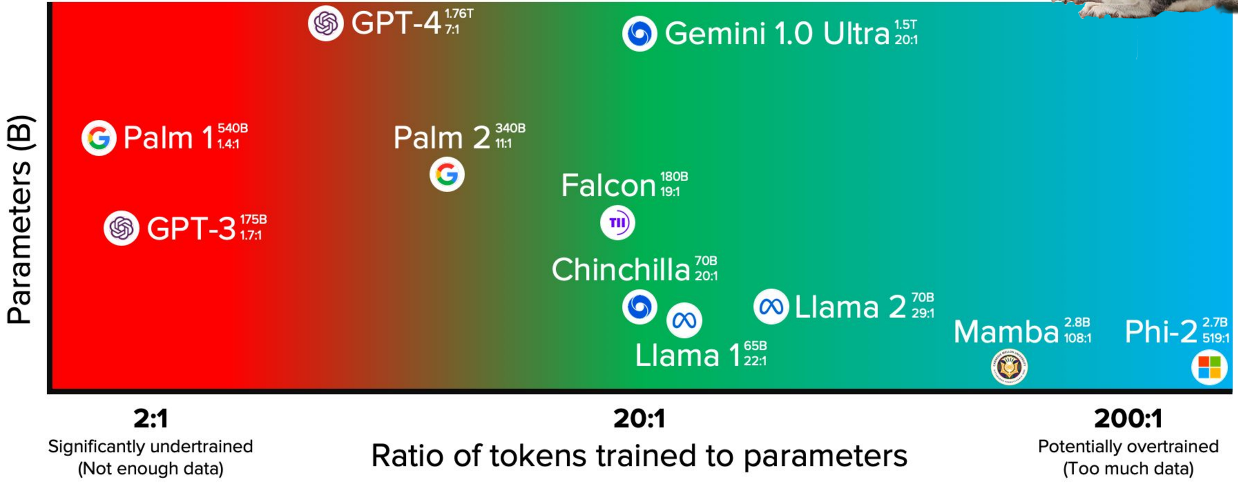
Scale Wisely: Chinchilla Rule

sources: https://arxiv.org/pdf/2203.15556
https://lifearchitect.ai/chinchilla/

Larger models perform better, but what size is enough for me?

It's often better to prioritize data scale over model scale.

Chinchilla rule of the thumb: **20 tokens per parameter**.



Selected highlights only. Mostly to scale. Informed estimates for Palm 2, GPT-4, and Gemini. Alan D. Thompson. November 2022, major update December 2023. https://lifearchitect.ai/

Try It Yourself: QLoRA

Credit: **Ondřej Plátek**

There are more tunable parameters than in regular Fine-Tuning, i.e. tunable layers, decomposition rank (quite robust), update scaling.

Hint: QLoRA defaults are usually good to start with.

https://github.com/artidoro/qlora

Try It Yourself: QLoRA

Credit: **Ondřej Plátek**

There are more tunable parameters than in regular Fine-Tuning, i.e. tunable layers, decomposition rank (quite robust), update scaling.

Hint: QLoRA defaults are usually good to start with.

Hint: Try running inference on quantized (but not adapted) model to see if the performance deteriorates.

https://github.com/artidoro/qlora

Try It Yourself: QLoRA

Credit: **Ondřej Plátek**

There are more tunable parameters than in regular Fine-Tuning, i.e. tunable layers, decomposition rank (quite robust), update scaling.

Hint: QLoRA defaults are usually good to start with.

Hint: Try running inference on quantized (but not adapted) model to see if the performance deteriorates.

Hint: Newer models are usually “quantization friendlier”

https://github.com/artidoro/qlora

Questions?