

# 理解 LoRA

<https://github.com/microsoft/LoRA>

该仓库包含最小化 LoRA 实现, 以及论文 *LoRA: Low-Rank Adaptation of Large Language Models* 作者 Edward J. Hu 对 LoRA 的视频解读。

## 模型与显存占用

Transformer Math 101 <https://blog.eleuther.ai/transformer-math/>

Total memory = model size + kv-cache + activation memory + optimizer/grad memory + cuda

1. Model size = 这是 .bin 文件大小(如果是 Q8 量化, 除以 2; 如果是 Q4 量化, 除以 4)。
2. KV-Cache = KV(键值) 向量占用的内存。大小 = 每层 (2 x 序列长度 x 隐藏层大小)。对于 Huggingface, 这是每层 (2 x 2 x 序列长度 x 隐藏层大小)。在训练中, 整个序列一次性处理(因此 KV 缓存内存 = 0)。
3. Activation Memory = 在前向传播中, 每个操作的输出都需要存储以便进行 .backward()。例如, 如果您执行  $output = Q * input$ , 其中  $Q = (dim, dim)$ ,  $input = (batch, seq, dim)$ , 那么形状为 (batch, seq, dim) 的输出需要存储(以 fp16 格式)。这在 LoRA/QLoRA 中消耗了最多的内存。在大型语言模型中, 有许多这样的中间步骤(在 Q、K、V 之后, 在注意力之后, 在规范化之后, 在 FFN1、FFN2、FFN3 之后, 在跳层之后...)。每层大约存储 15 个中间表示。
4. Optimizer/Grad memory = 梯度张量与优化器相关的张量占用的内存。
5. Cuda etc. overhead = 每次加载 cuda 时, CUDA 占用大约 500-1GB 的内存。此外, 当您使用任何量化(如 bitsandbytes)时, 会有额外的开销。这里没有直接的公式(我在计算中假设 CUDA 开销为 650 MB)。

Can it run llm <https://huggingface.co/spaces/Vokturz/can-it-run-llm>

Access token

Model name (Press Enter to apply)

google/gemma-7b

GPU Vendor

NVIDIA

Filter by RAM (GB)

10.0040.00

0.5096.00

GPU

A10 PCIe

LoRa % trainable parameters

2.00

0.10100.00

	INFO
Product Name	A10 PCIe
GPU Chip	GA102
Released	Apr 12th, 2021
Bus	PCIe 4.0 x16
Memory	24 GB, GDDR6, 384 bit
GPU clock	885 MHz
Memory clock	1563 MHz
Shaders / TMUs / ROPs	9216 / 288 / 96
RAM (GB)	24.0
Vendor	NVIDIA

Can you run it? LLM version

Information

- GPU information comes from [TechPowerUp GPU Specs](#)
- Mainly based on [Model Memory Calculator](#) by [hf-accelerate](#) using [transformers](#) library
- Inference is calculated following [EleutherAI Transformer Math 101](#), where is estimated as
$$\text{Memory}_{\text{Inference}} \approx \text{Model Size} \times 1.2$$
- For LoRa Fine-tuning, I'm assuming a 16-bit dtype of trainable parameters. The formula (in terms of GB) is
$$\text{Memory}_{\text{LoRa}} \approx \text{Model Size} + \left( \# \text{ trainable Params}_{\text{billions}} \times \frac{16}{8} \times 4 \right) \times 1.2$$

google/gemma-7b (8.6B)

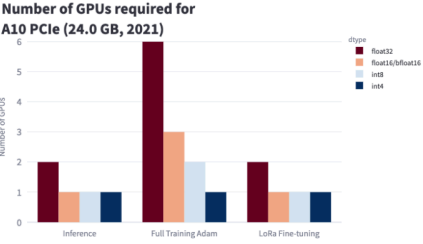
int4int8float16/bfloat16float32

✓ You require 1 GPUs for Inference

🚫 You require 3 GPUs for Full Training Adam

✓ You require 1 GPUs for LoRa Fine-tuning (2.0%)

	float32	float16/bfloat16	int8	int4
Total Size (GB)	31.81	15.9	7.95	3.98
Inference (GB)	38.17	19.08	9.54	4.77
Training using Adam (GB)	127.22	63.61	31.81	15.9
LoRa Fine-Tuning (GB)	39.81	20.72	11.18	6.41

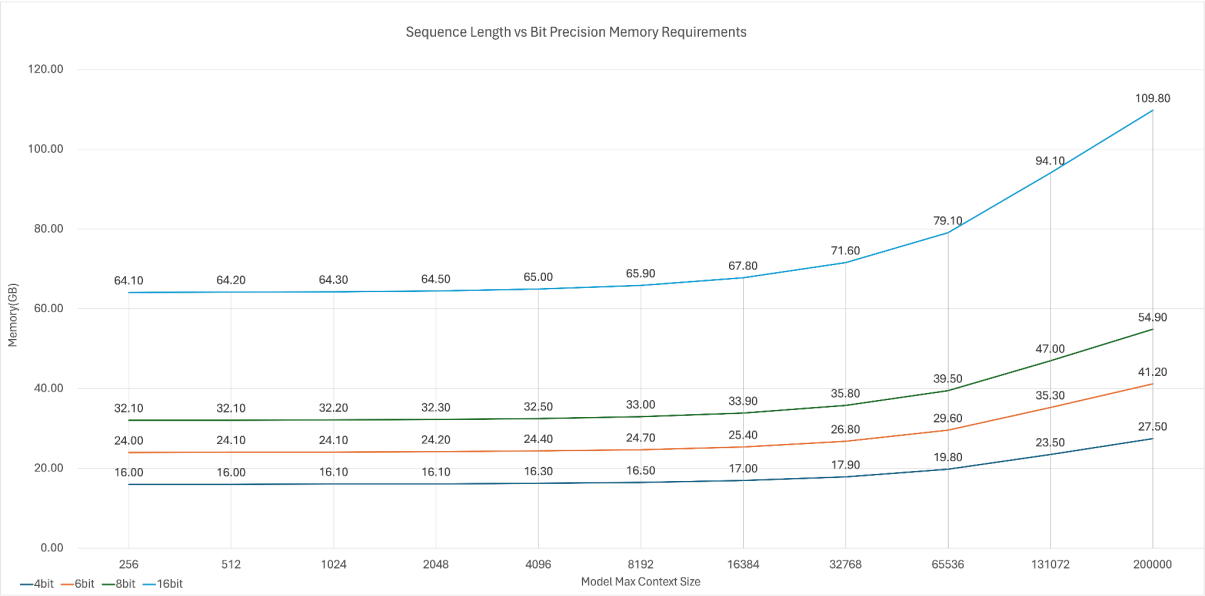


上下文长度与显存占用

quadratic complexity(二次计算复杂度)

Model: 34B-200K

Mode: infer



## 优化

- *FlashAttention 2* <https://github.com/Dao-AILab/flash-attention>
- *unsloth* <https://github.com/unslothai/unsloth>
- *Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention*  
<https://arxiv.org/abs/2404.07143>

## 长上下文与短上下文模型选择

- 准确度优先的任务 (例如function call) 选择短上下文模型 (例如4k)
- 可以接受的部分内容遗忘 (例如长文本总结任务), 选择长上下文模型

## Loss-in-the-Middle 问题

<https://arxiv.org/abs/2307.03172>

- 不进行微调的情况下, 将最相关的内容安排在两侧, 不太相关的内容安排在中间
  - 微调, 将与问题相关的内容均匀的分布在整个上下文中
- <https://github.com/microsoft/FILM>