Training Large Language Models

Memory and Compute Requirements

Learning goals

- Learn about different contributions to compute requirements
- Learn how model size components influence memory requirements

COMPUTE REQUIREMENTS

Basic equation: Cost to train a transformer (decoder) model:

$$C \approx \tau T = 6PD$$

▸ Source: Quentin et al., 2023

COMPUTE REQUIREMENTS

where:

- C: No. of floating-point operations (FLOPs) to train the model: $C = C_{forward} + C_{backward}$
- $C_{forward} \approx 2PD$
- $C_{backward} \approx 4PD$
- τ is throughput of hardware: (No. GPUs) x (FLOPs/GPU)
- T is the time spent training the model, in seconds
- P is the number of parameters in the model
- D is the dataset size (in tokens)

COMPUTE UNITS

C can be measured in different units:

- FLOPs-seconds which is [Floating Point Ops / Second] x [Seconds]
  - We also use multiples GFLOP-seconds, TFLOP-seconds etc.
  - Other multiples like PFLOP-days are used in papers
  - 1 PFLOP-day = $10^{15} \cdot 24 \cdot 3600$ FLOP-seconds
- GPU-hours which is [No. GPUs] x [Hours]
  - GPU model is also required, since they have different compute capacities
  - For any GPU model, its Actual FLOPs are always lower than the advertised theoretical FLOPs

PARAMETER VS DATASET

- Model performance depends on number of parameters P, but also on number of training tokens D
- We need to decide about P and D, so that we get the best performance withing the compute budget
- The optimal tradeoff between P and D is: $D = 20P$
  - This is usually true for Chinchilla models ▸ Hoffmann et al., 2022, but not for all LLMs
- Training a LLM for less than 200 billion tokens is not recommended
- For models used in production: First determine the acceptable maximum inference cost, and then train the biggest model within that budget.

MEMORY REQUIREMENTS

Common questions:

- How big is this model in bytes?
- Will it fit/train in my GPUs?

Model size components:

- Model parameters
- Optimizer states
- Gradients
- Activations

NUMBER REPRESENTATIONS

- Pure fp32: single precision floating point number as defined by ▸ IEEE 754 standard, takes 32 bits or 4 bytes
- fp16: half precision float number as defined by ▸ IEEE_754-2008, occupying 16 bits or 2 bytes
- bf16 or brain floating point 16, developed by Google Brain project, occupying 16 bits or 2 bytes
- int8: integer from -128 to 127, occupying 8 bits or 1 byte

MODEL PARAMETERS

Parameter size depends on chosen representation:

- Pure fp32: $Mem_{model} = 4\ bytes/param \cdot N_{params}$
- fp16 or bf16: $Mem_{model} = 2\ bytes/param \cdot N_{params}$
- int8: $Mem_{model} = 1\ byte/param \cdot N_{params}$

It is practically common to use mixed representations:

- fp32 + fp16
- fp32 + bf16

OPTIMIZER STATES

AdamW: $Mem_{\text{optimizer}} = 12\ bytes/param \cdot N_{params}$

- fp32 copy of parameters: 4 bytes/param
- Momentum: 4 bytes/param
- Variance: 4 bytes/param

bitsandbytes (8-bit optimizer): $Mem_{\text{optimizer}} = 6\ bytes/param \cdot N_{params}$

- fp32 copy of parameters: 4 bytes/param
- Momentum: 1 byte/param
- Variance: 1 byte/param

SGD-like optimizers with momentum:
$Mem_{\text{optimizer}} = 8\ \text{bytes}/(\text{param} \cdot N_{params})$,

- fp32 copy of parameters: 4 bytes/param
- Momentum: 4 bytes/param

GRADIENTS

They are usually stored in the same datatype as the model parameters.

Their memory overhead contribution is:

- fp32: $Mem_{grad} = 4\ bytes/param \cdot N_{params}$
- fp16 or bf16: $Mem_{grad} = 2\ bytes/param \cdot N_{params}$
- int8: $Mem_{grad} = 1\ byte/param \cdot N_{params}$

ACTIVATIONS

- GPUs are bottlenecked by memory, not FLOPs
- Save GPU memory by recomputing activations of certain layers
- Various schemes for selecting which layers to clear
- They take some extra memory, but save even more

Total memory when training **without** activations:
$Mem_{training} = Mem_{params} + Mem_{opt} + Mem_{grad}$

Total memory when training **using** activations:
$Mem_{training} = Mem_{params} + Mem_{opt} + Mem_{grad} + Mem_{activ}$

In the latter case, Mem_{params}, Mem_{opt} and Mem_{grad} are significantly smaller than in the former.