LLM Efficiency

Tomasz Limisiewicz

18 April 2024
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
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: $O(n^2 \cdot d)$
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: $O(n^2 \cdot d + n \cdot d^2)$
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 $\rightarrow$ 20GB

Size of dataset:

Pre-training: ~10T tokens $\rightarrow$ 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
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

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!

PaLM

BLOOM

YaLM

GPT-NeoX

GPT-2
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
Weight matrices are decomposed

\[ W \in \mathbb{R}^{d \times d} \]

\[ A = \mathcal{N}(0, \sigma^2) \]

\[ B = 0 \]

\[ r \]

\[ d \]

\[ x \]

\[ h \]
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:

\[ h = W_0 x + \Delta W x = W_0 + BA \]

Where

\( W_0 \in \mathbb{R}^{d \times k} \)

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

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

source: https://arxiv.org/pdf/2106.09685
Fine-Tuning vs. LoRA

Full Finetuning (No Adapters)

Optimizer State (32 bit)

Adapters (16 bit)

Base Model

16-bit Transformer

LoRA

Parameter Updates

Gradient Flow

source: https://arxiv.org/pdf/2305.14314
The size of parameters may be decreased by quantization.

Parameters are assigned into coarse buckets.

Important to determine the range of the quantization $c$. 

Quantization
4bit NormalBit quantization: equally-sized buckets based
4bit NormalBit quantization: equally-sized buckets based

Double Quantization: quantize both parameters but also their range $c$
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

Optimizer weights are transferred between GPU and CPU memory.

It prevents running out of memory when processing long sequences.

source: https://arxiv.org/pdf/2305.14314
Fine-Tuning vs. LoRA vs. QLoRA

### Fine-Tuning vs. LoRA vs. QLoRA

<table>
<thead>
<tr>
<th></th>
<th>Fine-Tuning</th>
<th>LoRA</th>
<th>QLoRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tunable parameters</td>
<td>100 %</td>
<td>~0.1%</td>
<td>~0.2%</td>
</tr>
<tr>
<td>Model Precision</td>
<td>16 bit</td>
<td>16 bit</td>
<td>4 bit</td>
</tr>
<tr>
<td>RAM 10B model</td>
<td>160GB</td>
<td>~40GB</td>
<td>~12GB</td>
</tr>
<tr>
<td>Applicable for</td>
<td>Industrial Supercomputer</td>
<td>Academic Cluster</td>
<td>Good Personal Setting</td>
</tr>
<tr>
<td>Matches performance</td>
<td>—</td>
<td>YES</td>
<td>YES*    (in full model tuning)</td>
</tr>
</tbody>
</table>
Larger models perform better, but what size is enough for me?

It’s often better to prioritize data scale over model scale.
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.**
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.

**Try It Yourself: QLoRA**

[https://github.com/artidoro/qlora](https://github.com/artidoro/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
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?