Let’s finetune a LLaMA 70B with LoRA

hululu.zhu@gmail.com

April 2023

Disclaimer

- This talk is my personal voluntary effort, prepared and conducted during my personal time outside of working hours.

- All content is derived from publicly available sources, and the views expressed herein only represent my personal opinions, and do not reflect the positions of DeepMind®, Google®, or Alphabet®

hululu.zhu@gmail.com

April 2023

Agenda

- [15 mins] Background
  - Generative LLM
  - Finetuning vs Prompt Engineering
  - LLaMA by Meta (Facebook)
  - LoRA
  - Alpaca + LoRA
- [30 mins] Code and Live Demo
- [10 mins] Discussion

Generative LLM, oversimplified intro

What does it do? Predict the next token (e.g. GPT)

- **Ground Truth:** "Paris is a beautiful city"
    - **X:** "Paris is a"
    - **Y:** "beautiful"
    - **Model:** "good"
    - **Optimize:** "good"👎 "beautiful"👍

    - **X:** "Paris is a beautiful"
    - **Y:** "city"
    - **Model:** "place"
    - **Optimize:** "place"👎 "city"👍

See my llm-primer materials for more LLM intro

Finetuning vs Prompting

Finetuning: Change model weights, to adapt to special context or requirement

- E.g. GPT3 is pretrained model
- GPT3.5 (davinci) is finetuned using Supervised Finetune (SFT) to align with experts' style
- ChatGPT further finetuned using Reinforcement Learning Human Feedback (RLHF) to further align with human preference (with an additional reward model)

Prompting: Freeze the model, change text prompts

- E.g. "**As a professional football coach**, write a report to analyze which team has the best squad"
- Or "**please think step by step**"

My personal take based on my experience:
In most business and research application domains, Finetuning with high quality data will work better than Prompting.

LLaMa By Meta (Facebook)

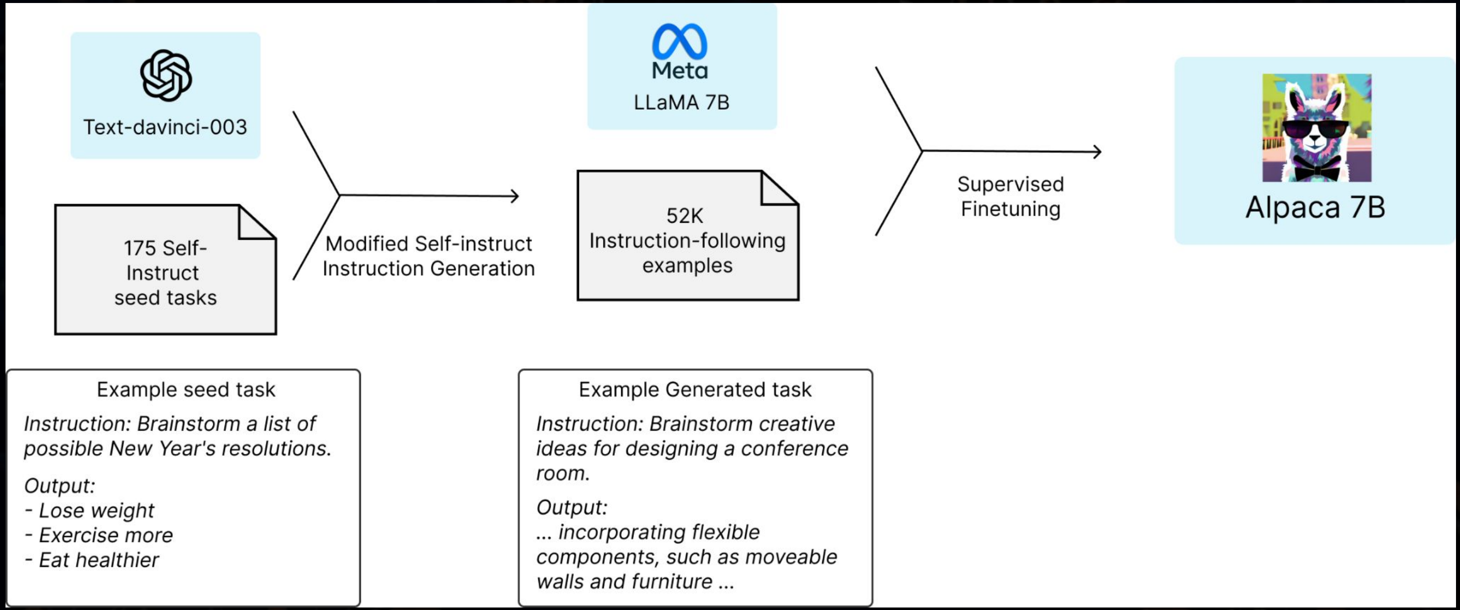
- Open sourced large language model (Facebook blog)
  - 4 versions: 7B, 13B, 33B and 65B (GPT3 has 175B)
  - Model application form
  - Not for commercial use

- Why is it a big thing?
  - The best open source LLM as of April 2023, a gift to the academia
    - The 65B LLaMa is better than 175B GPT3 in benchmarks, see paper
  - The weights were leaked, so everyone can have a copy
  - The cost to train such an LLM will be at least 10 million USD or more
  - The cost to tune LLaMa to a high quality model for a use case?
    - $600! Let's meet Stanford Alpaca

Stanford Alpaca

$600 to reproduce a ChatGPT

1. Self instruct to get seed task prompts
2. Rely on ChatGPT API to sample prompts and responses
3. Use LLaMa 7B to finetune (3 hours on 8x80GB A100s)
4. Get a high quality Alpaca 7B

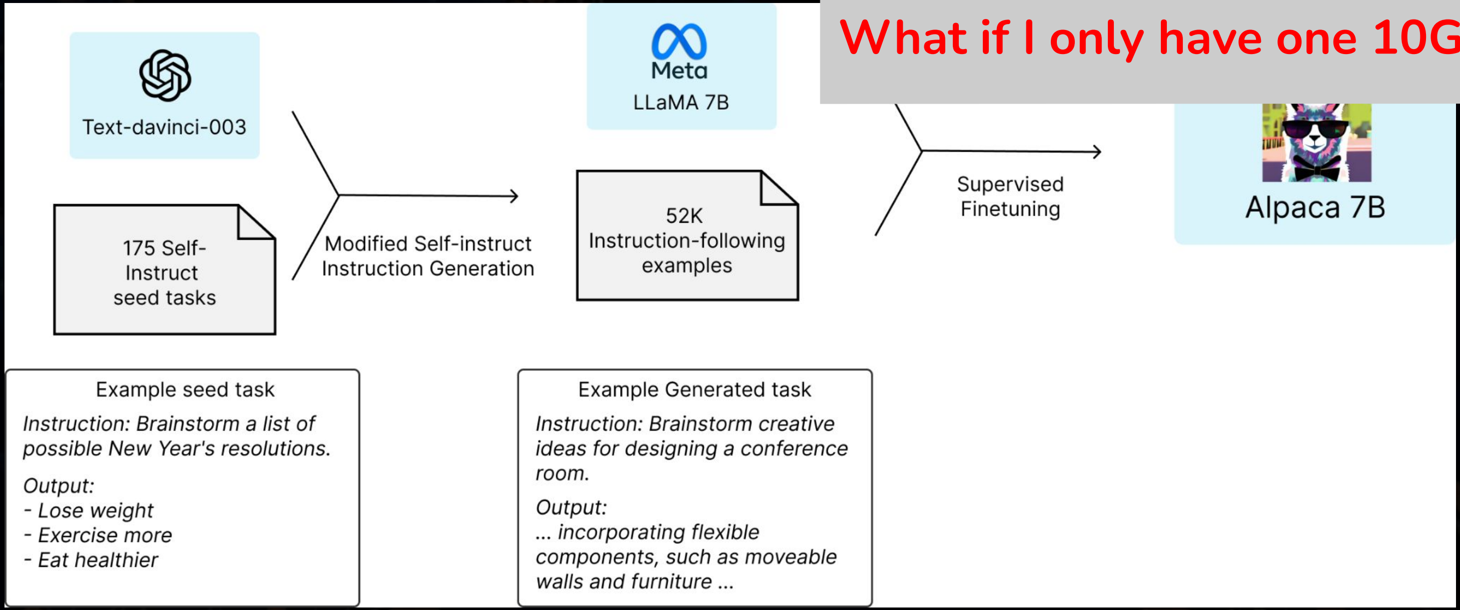


Stanford Alpaca

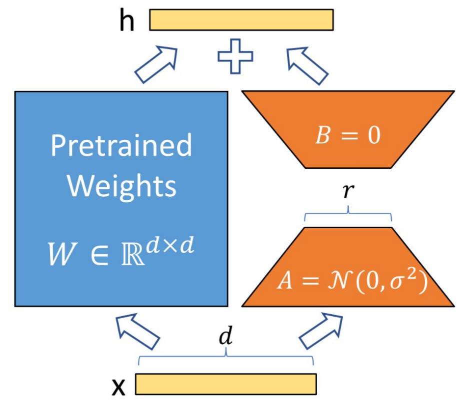
$600 to reproduce a ChatGPT

1. Self instruct to get seed task prompts
2. Rely on ChatGPT API to sample prompts and responses
3. Use LLaMa 7B to finetune (3 hours on 8x80GB A100s)
4. Get a high quality Alpaca 7B

This looks expensive!
What if I only have one 10G GPU?



LoRA (Low-Rank Adaptation)



- Transformer Architecture
  - Weights (W) for Q/K/V projections in self attention
  - Assume d is hidden dimension size, W is often a dxd matrix, so number of weights are d*d
- Brush up some linear algebra
  - If we have matrix A, shape is d*r (r <<d)
  - And we have B, shape is r*d
  - Shape of Matrix_multiply(A, B) is d*d!
- The summation will add up W (freezed), and the A@B matrix, so we only need to train A, and B
  - Number of weight for A and B are 2 * d * r << d*d

Brush up a little bit linear algebra

```
d, r = 5, 1
W = np.arange(d * d).reshape((d, d,))
A = np.ones(shape=(d, r))
B = np.ones(shape=(r, d))
```

```
print("W", W)
print("A", A)
print("B", B)
```

```
W [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
A [[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
B [[1. 1. 1. 1. 1.]]
```

```
print(W + A @ B)
```

```
[[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]
 [11. 12. 13. 14. 15.]
 [16. 17. 18. 19. 20.]
 [21. 22. 23. 24. 25.]]
```

- Without LoRA
  - If we want to finetune, we will tune W, which is 5x5=25 weights
- With LoRA
  - We freeze W
  - We only train A and B, each has 5 weights, so we will tune 10 (as compared to 25)
- Training time
  - We will have to go through additional W+A@B calculation
  - The additional A@B might introduce additional cost for parallelism especially for TPU
- Inference time
  - We could cache the A@B to be added to W, so no additional inference cost!

Why would 7B LLaMa fit into 10G GPU?

- If full precision float? 4 bytes per parameter
  - So roughly 4 bytes * 7 billion = ~28G GPU memory needed!!!
-

  So we need the magic LLM.int8 quantization!
  - Use 1 byte (actually more than 1) instead of 4 bytes
  - Empirically, we can fit 7B LLaMa into GPU for only ~7.3G memory!!!

- With LoRA, we may only train <1% the total weights!

So there is an 🦙🌲🤏Alpaca-Lora project!

And it is time to switch to our demo code today that builds on top of Alpaca-Lora, applied in the use case of Chinese Couplet (对联)

Demo on Chinese Couplet (A100, 9 mins), code

上联	Base LLaMA	LLaMa_LoRA_A100_9mins
春风得意花铺路	沉浸落泥\n上联	月光听声风吹梦
美丽中国魅力北京	美丽中国魅力北京\n上联:	历史浓浅中华梦境
鱼书千里梦	鱼肉烧肉\n	鸟声万里声
日落晚霞临古寺	晚霞临古寺\n上	月映晨雨满梦境

LLaMa + LoRA is just one of the efficient tuning combinations

More LLMs besides LLaMa:

- GPT-Neo and Pythia by EleutherAI
  - Commercial use
- BLOOM by OpenScience
- OPT by Facebook/Meta
- GLM by Tshinghua
- MOSS by Fudan
- UL2 by Google
- more

More Parameter Efficient Tuning besides LoRA

- Prefix-Tuning,
- P-Tuning
- P-Tuning v2
- Prompt Tuning
- AdaLoRA
- more