unsloth

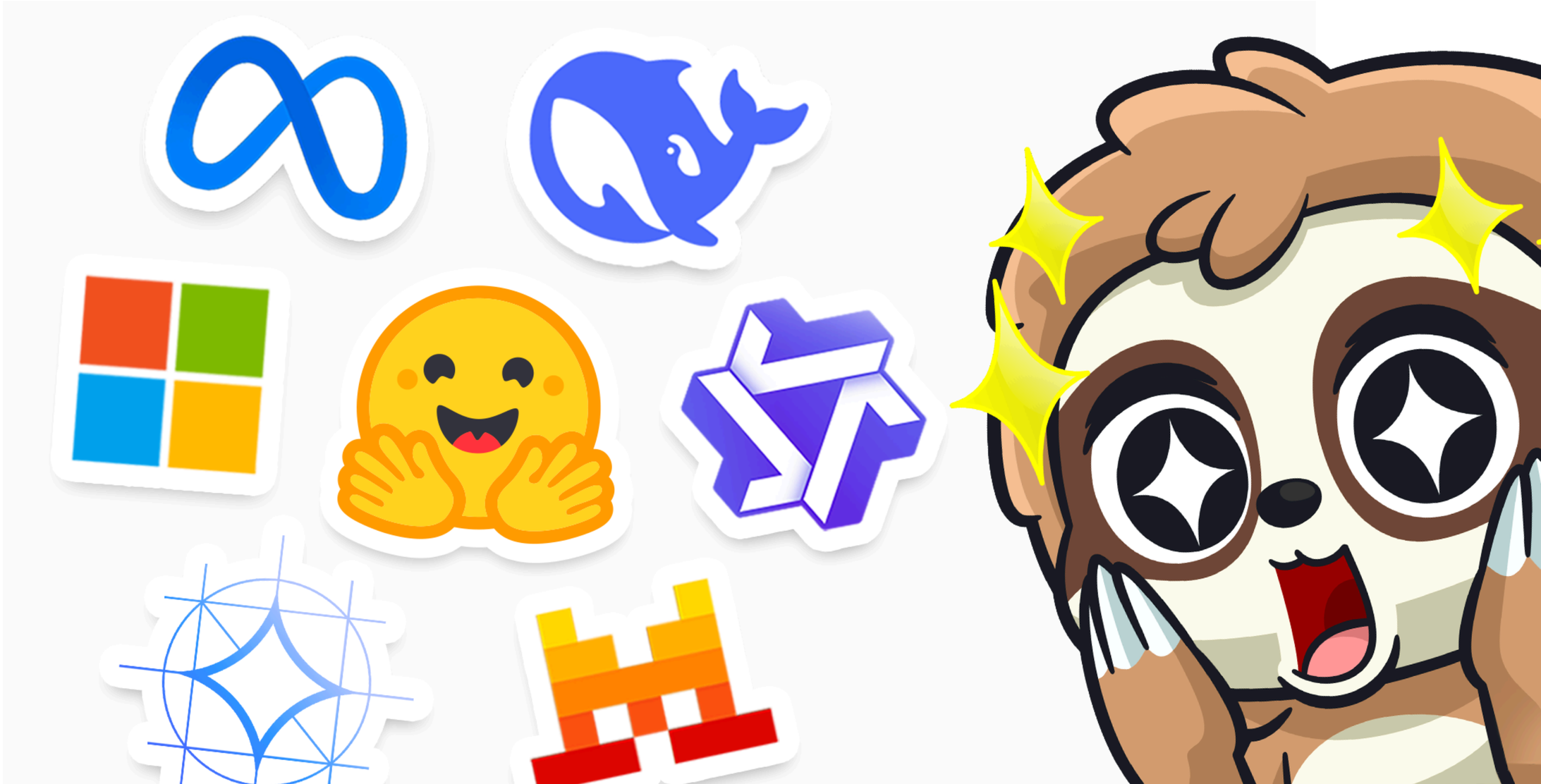
Maximizing Luck in Reinforcement Learning

Daniel Han

Co-founder

docs.unsloth.ai

Run & train LLMs locally



100M+

model downloads on
🤗Hugging Face

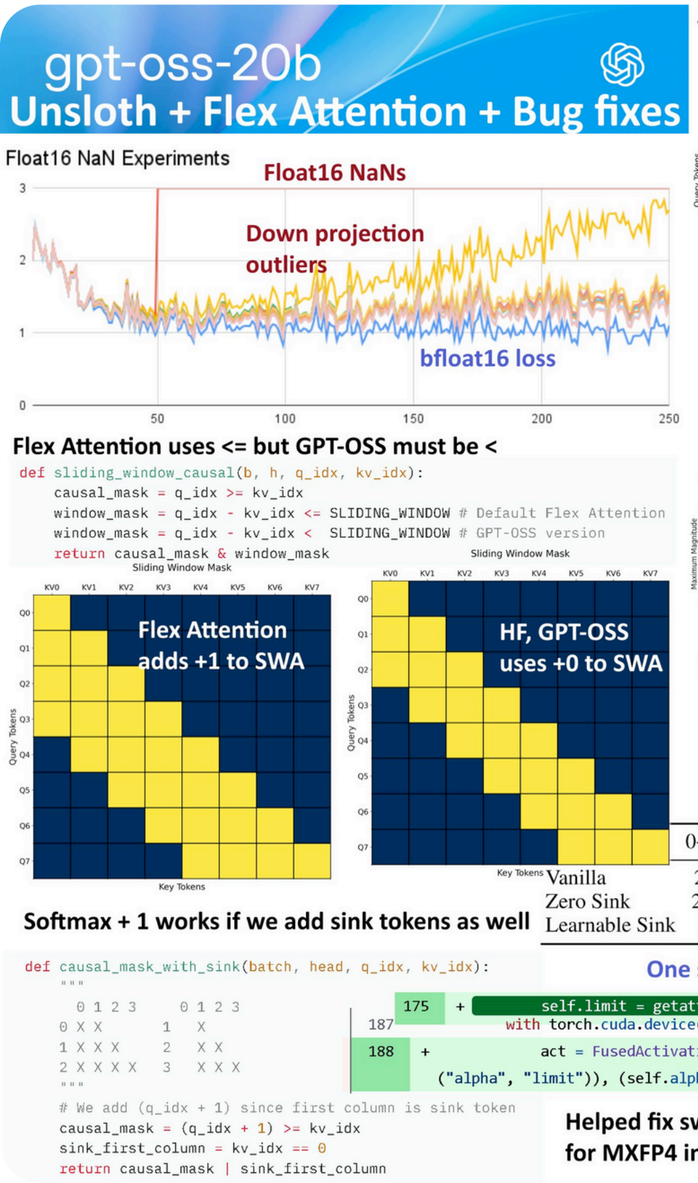
86K

models uploaded public to HF,
trained with Unsloth

47K

⭐stars on GitHub

unsloth



Gradient Acc Bug Fix Constant bug fixes for running LLMs

We worked with Hugging Face and OpenAI on gpt-oss.

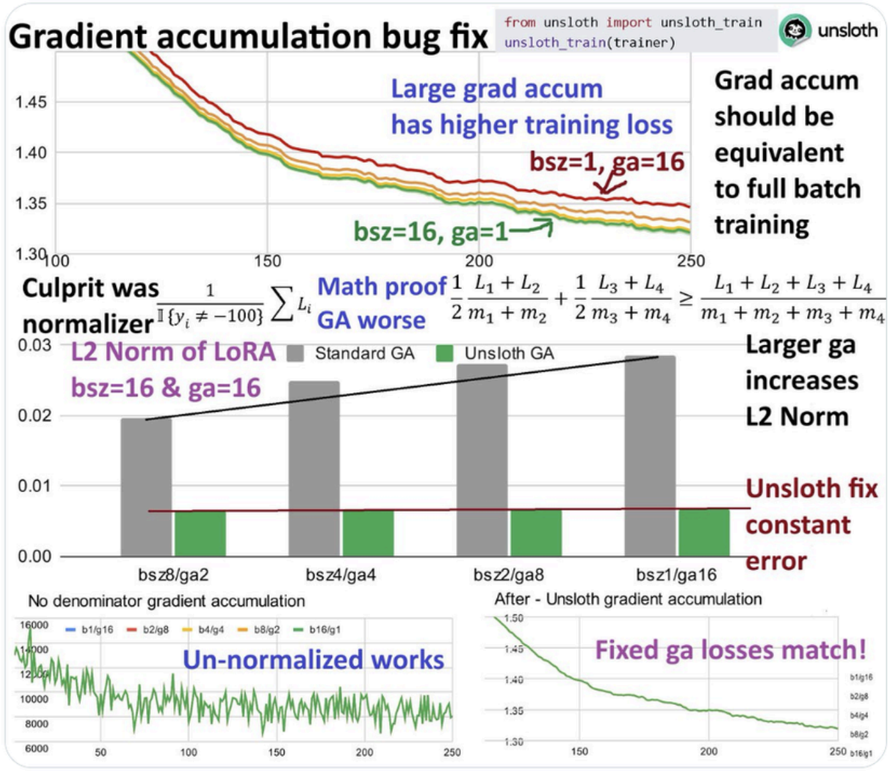
We work with HF, Google, Meta, Mistral to fix Gemma, Llama, Mistral, Phi bugs

Daniel Han @danielhanchen · Oct 15

Fixed a bug which caused all training losses to diverge for large gradient accumulation sizes.

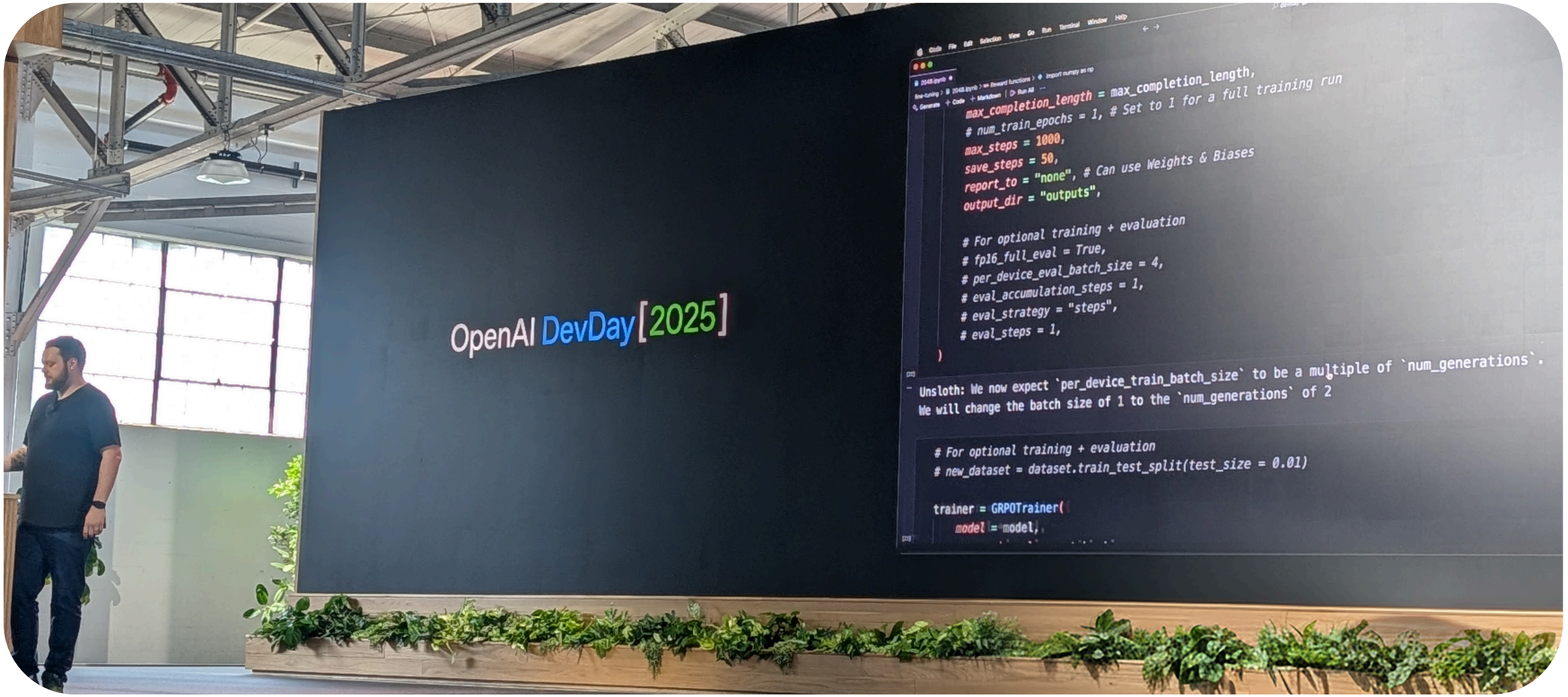
1. First reported by @bnjmn_marie, GA is supposed to be mathematically equivalent to full batch training, but losses did not match.
2. We reproed the issue, and further investigation

Show more



21 173 755 303K

OpenAI Reinforcement Learning 2048



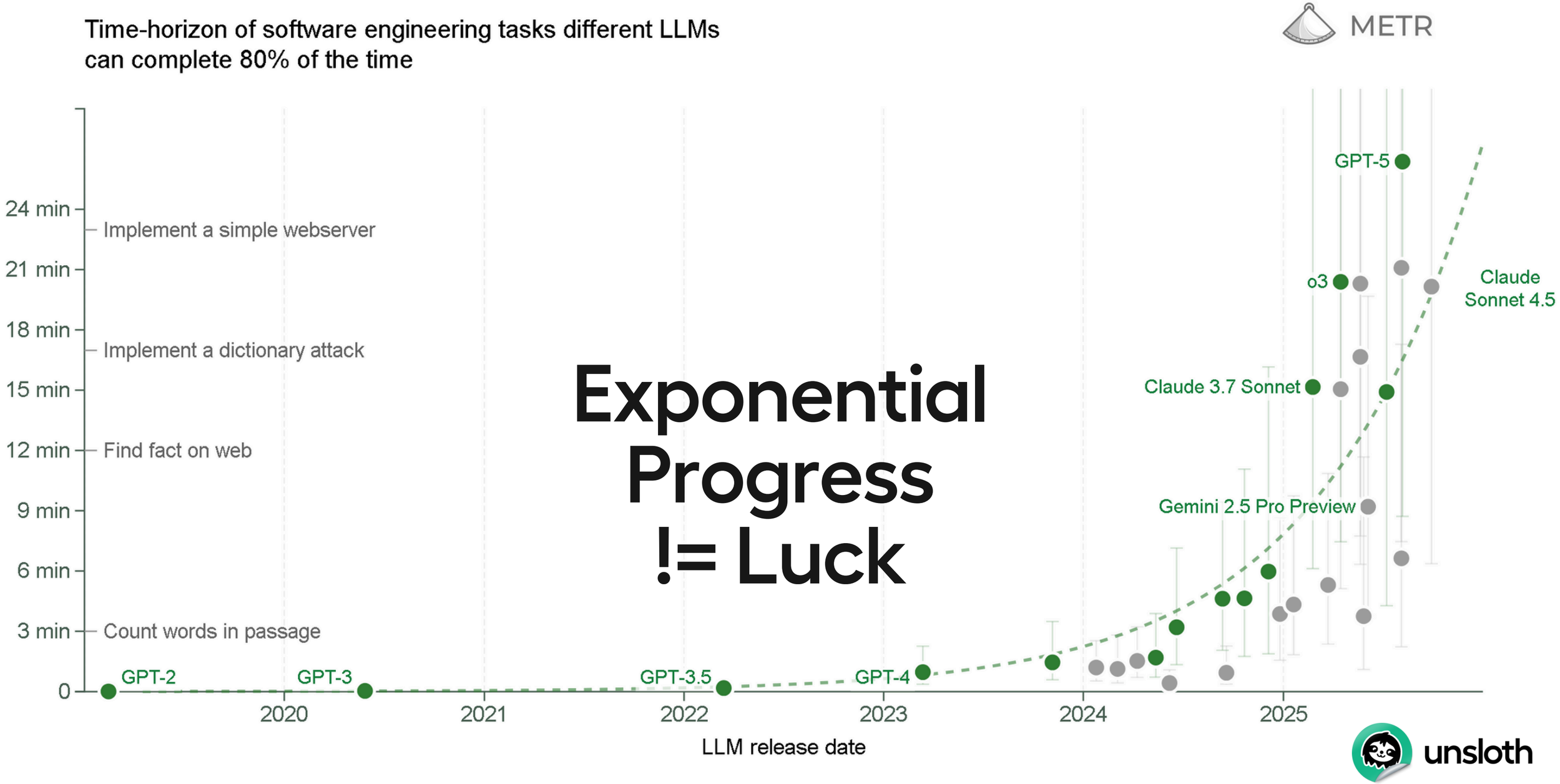
unsloth

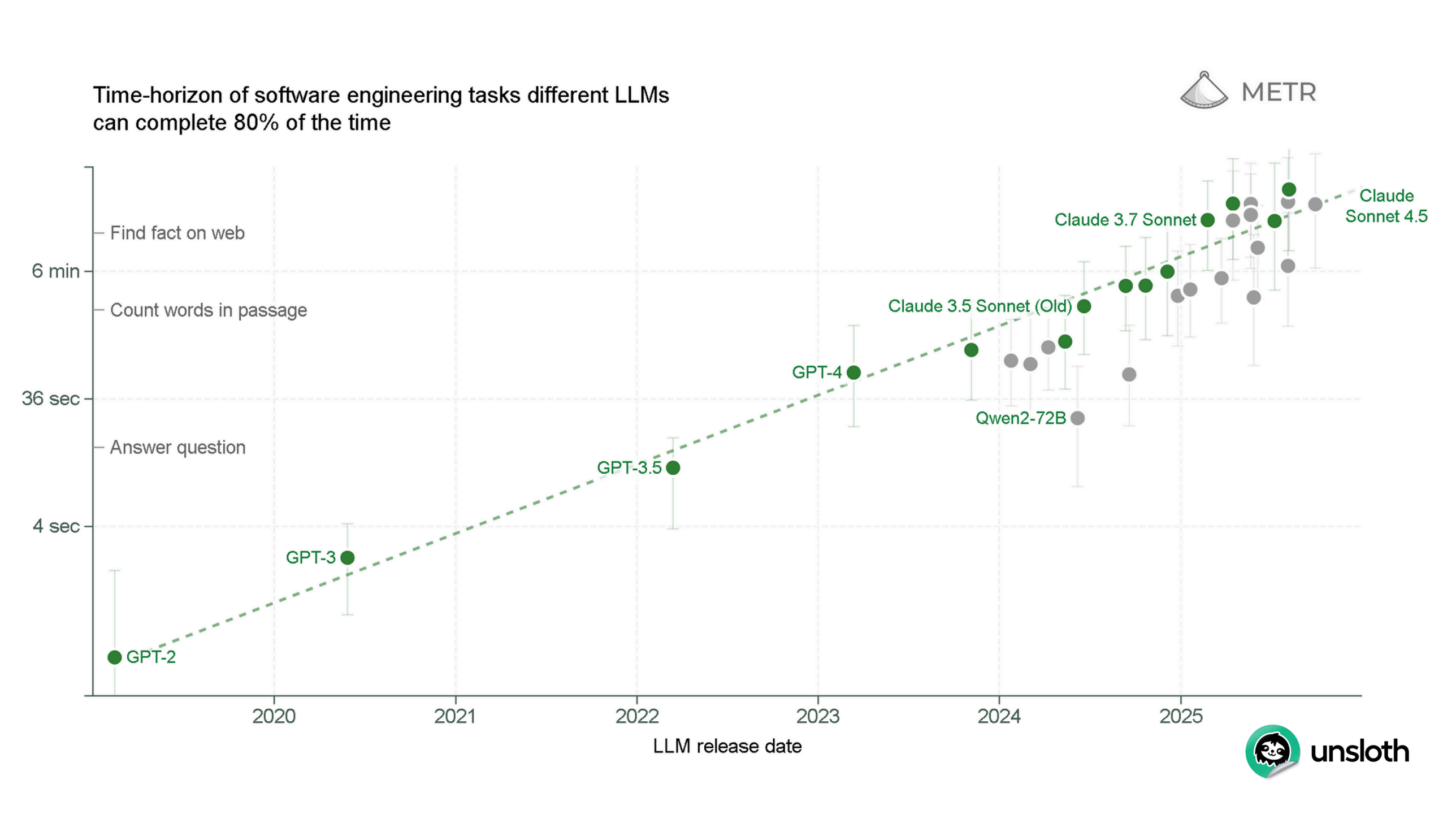
Maximizing Luck in Reinforcement Learning

What did I say Luck

Surely RL != Luck?

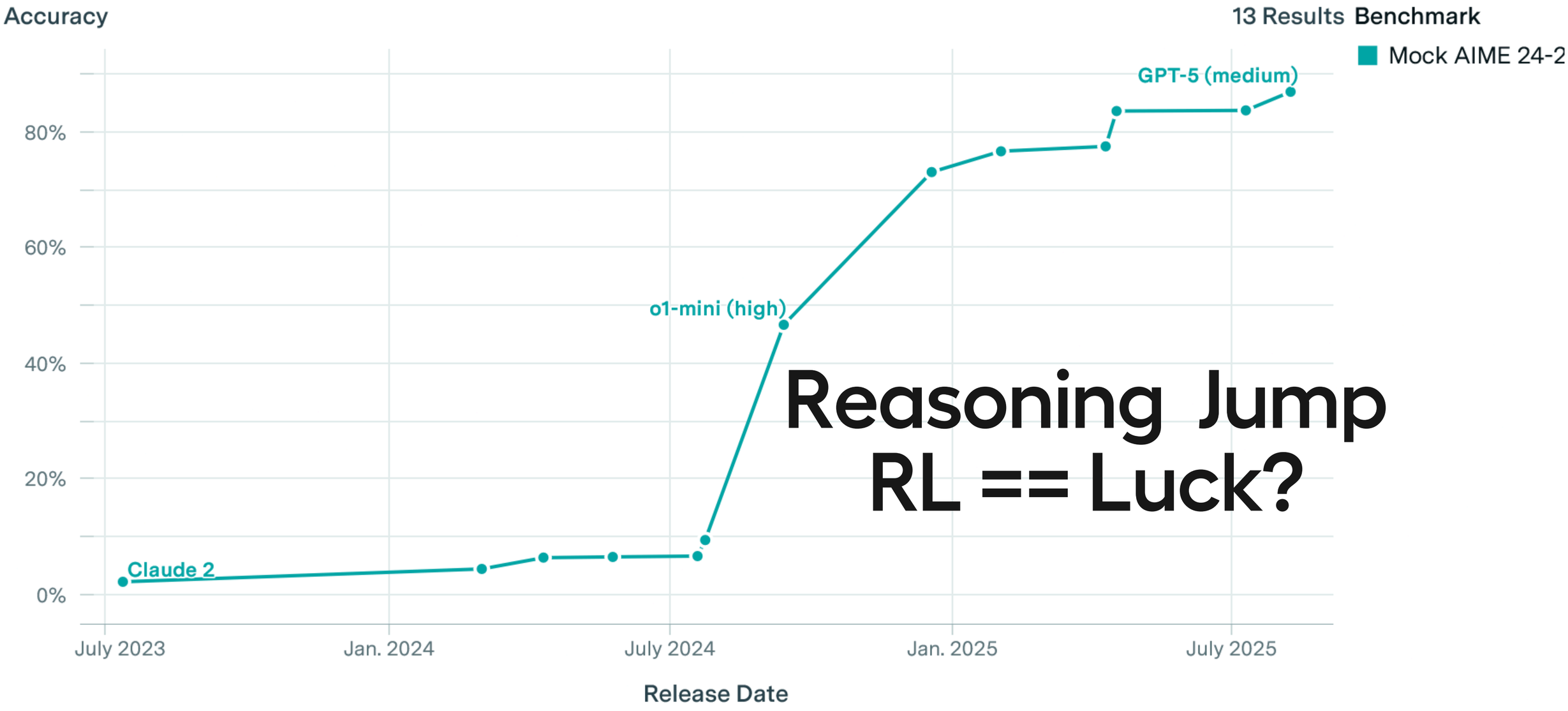
unsloth


Time-horizon of software engineering tasks different LLMs can complete 80% of the time
METR
24 min
21 min
18 min
15 min
12 min
9 min
6 min
3 min
0
Implement a simple webserver
Implement a dictionary attack
Find fact on web
Count words in passage
GPT-2
GPT-3
GPT-3.5
GPT-4
GPT-5
o3
Claude Sonnet 4.5
Claude 3.7 Sonnet
Gemini 2.5 Pro Preview
2020
2021
2022
2023
2024
2025
LLM release date
Exponential Progress != Luck
unsloth


Time-horizon of software engineering tasks different LLMs can complete 80% of the time
METR
Find fact on web
6 min
Count words in passage
36 sec
Answer question
4 sec
GPT-2
GPT-3
GPT-3.5
GPT-4
Claude 3.5 Sonnet (Old)
Qwen2-72B
Claude 3.7 Sonnet
Claude Sonnet 4.5
2020
2021
2022
2023
2024
2025
LLM release date
unsloth

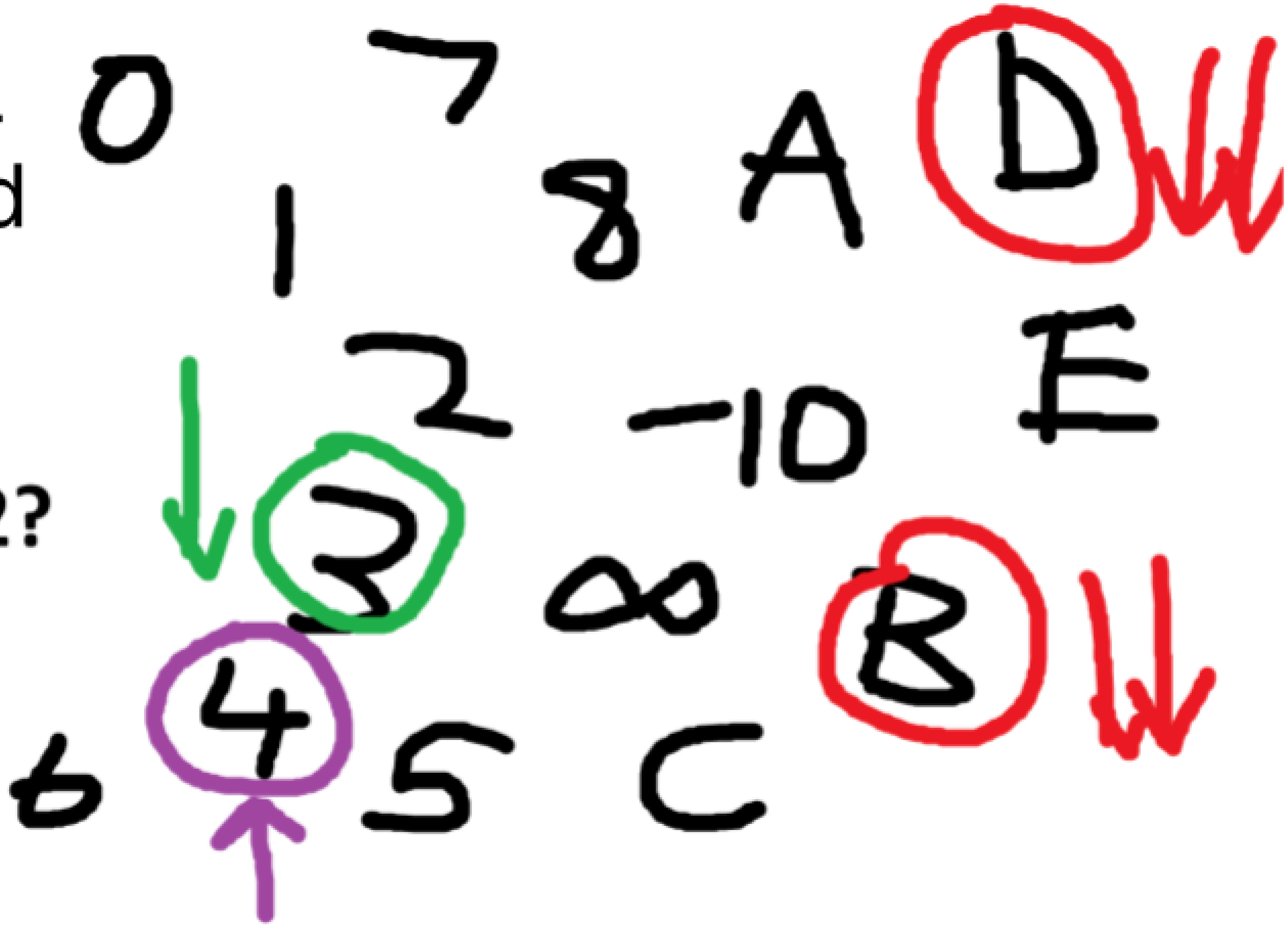
Frontier performance across benchmarks

EPOCH AI



Goal of RL
More good
Less bad

What is 2+2?



🤞 Luck (well Patience) Is All You Need

So I like to call it as "Luck Is All You Need" for RL.

Well a better phrase is "Patience is All You Need" for RL.

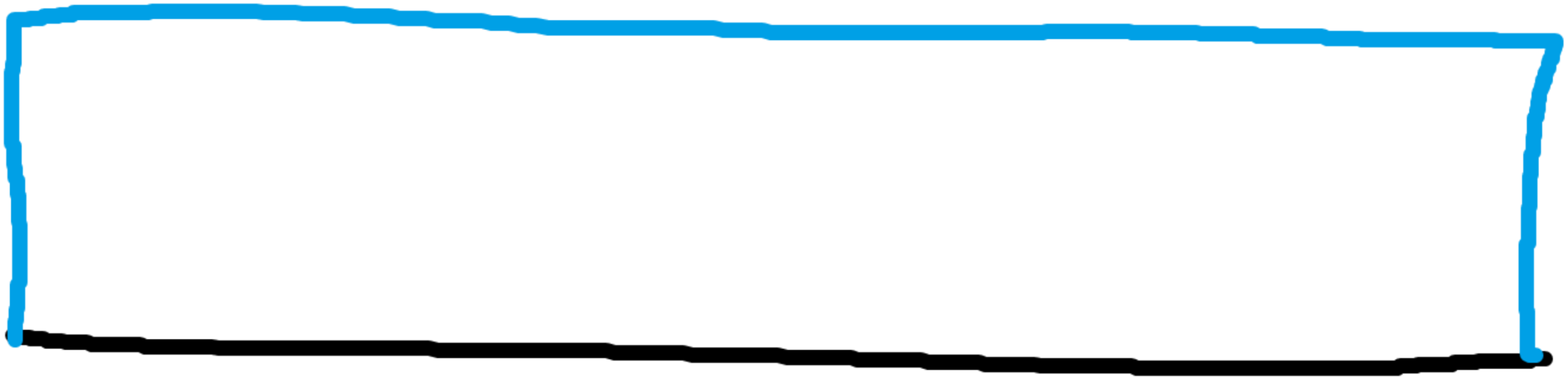
Patience is all you need

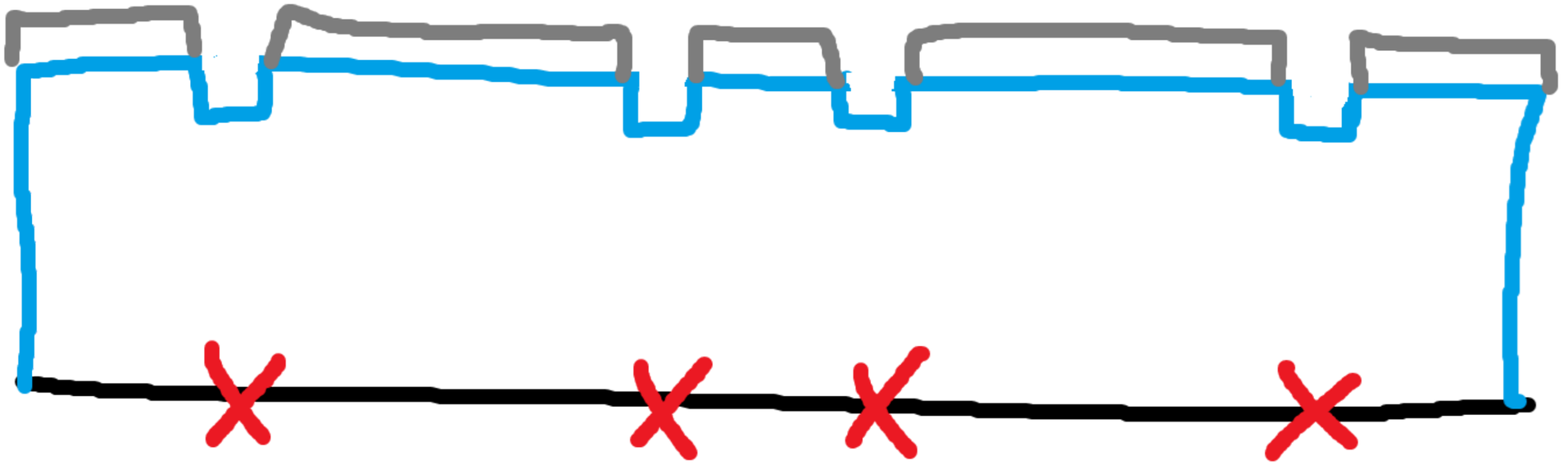
RL is not “dumb”

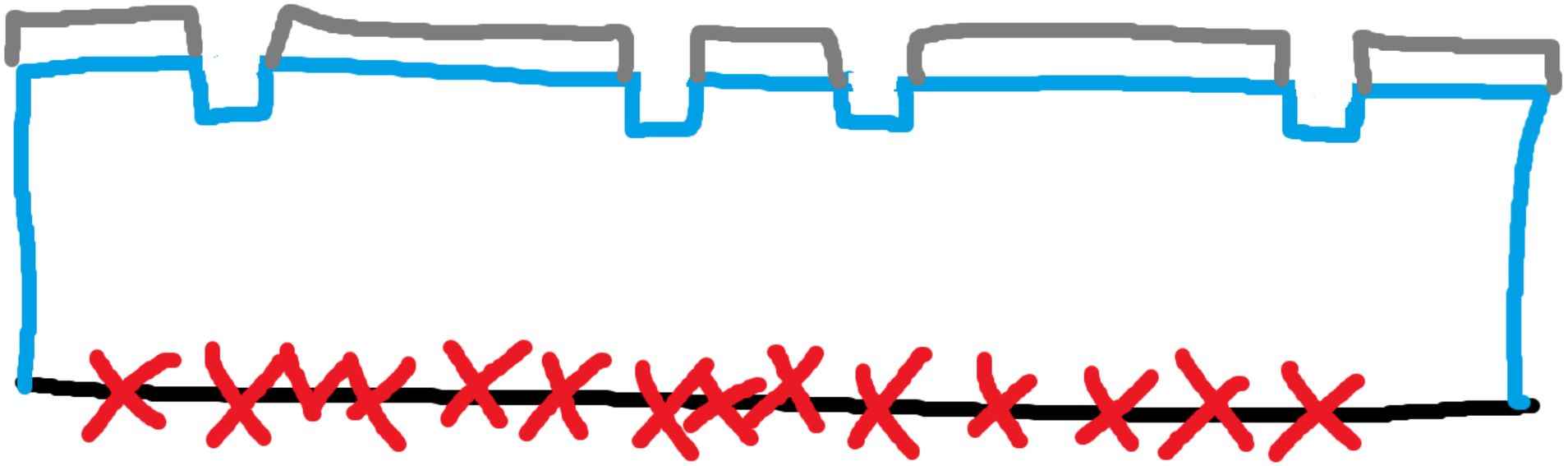
“RL is terrible; everything else is much worse”

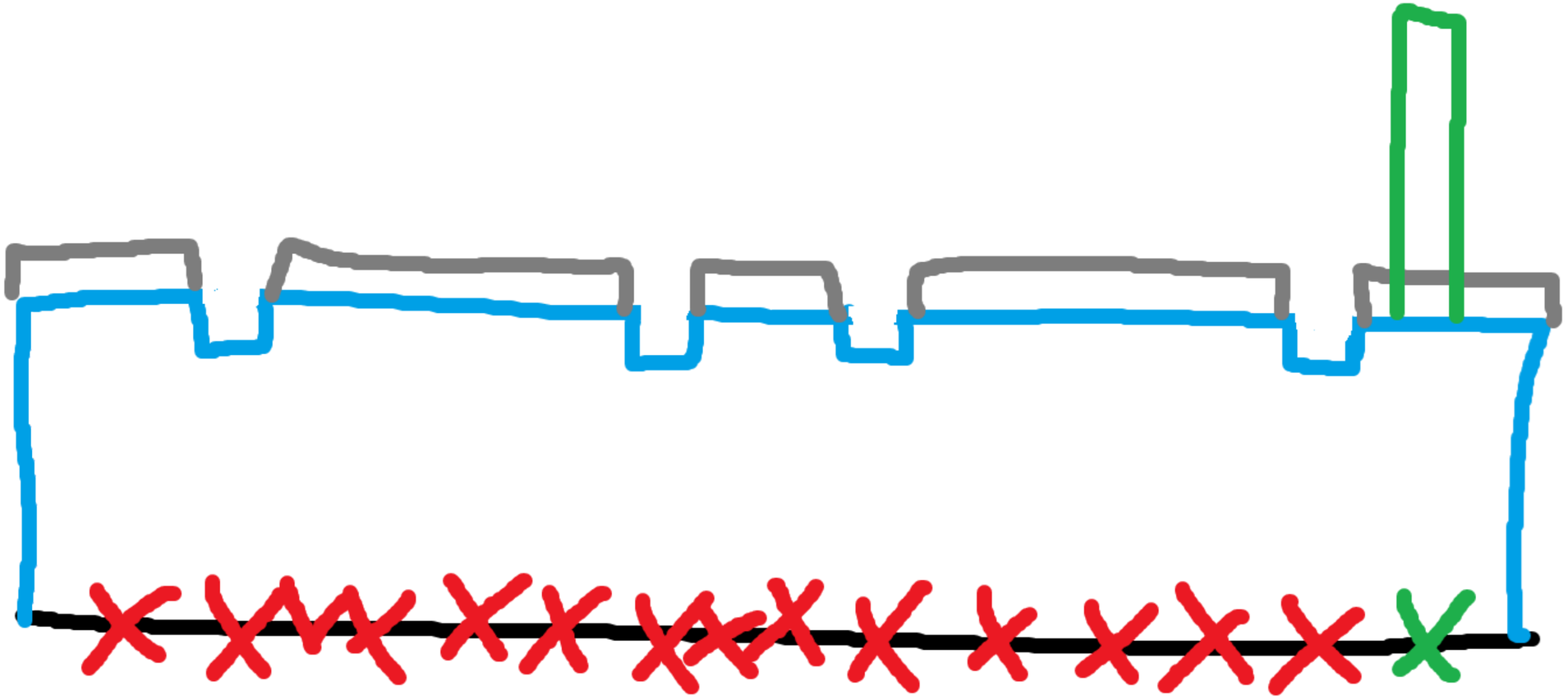
Andrej Karpathy, Dwarkesh Podcast

unsloth



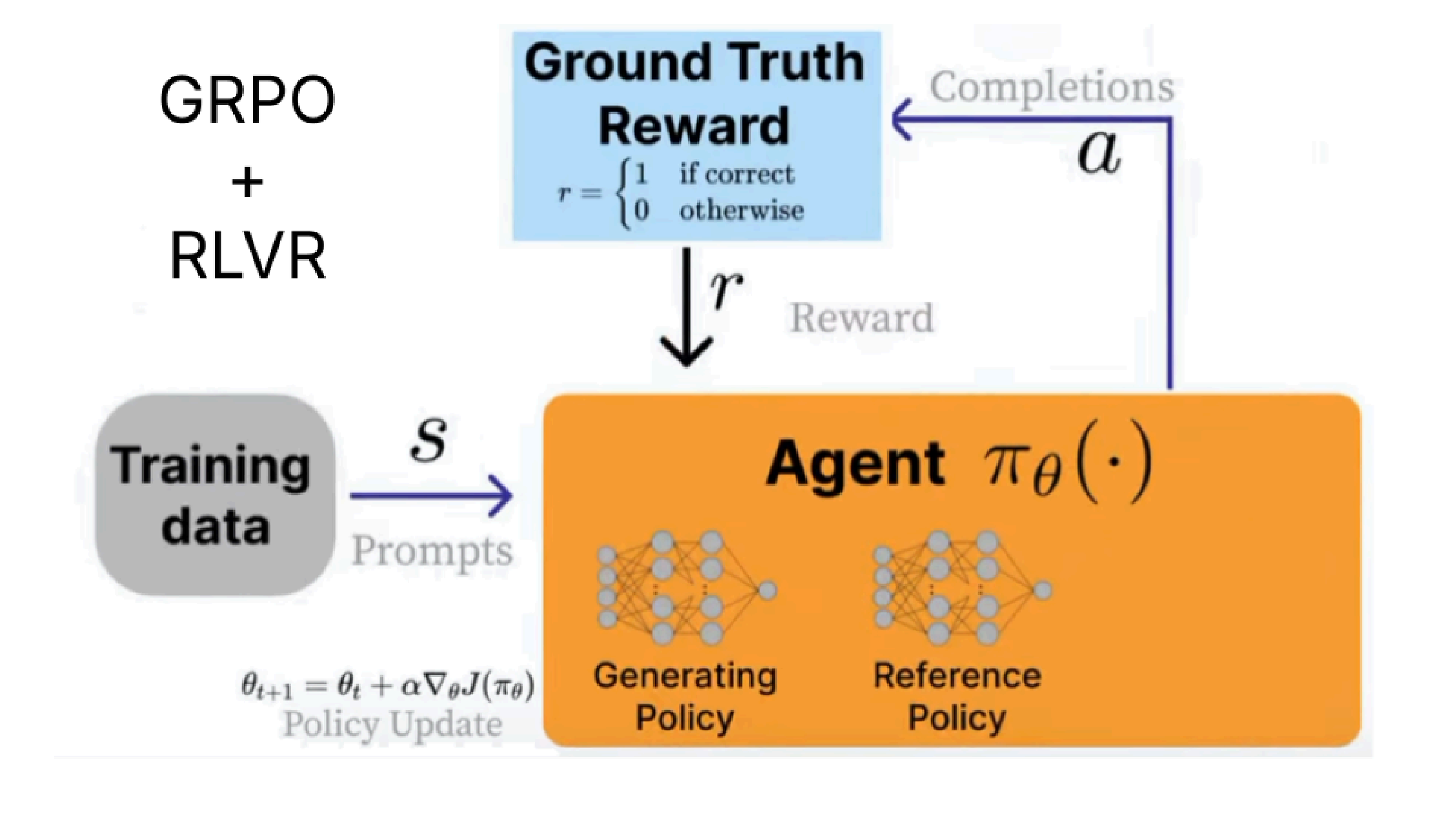


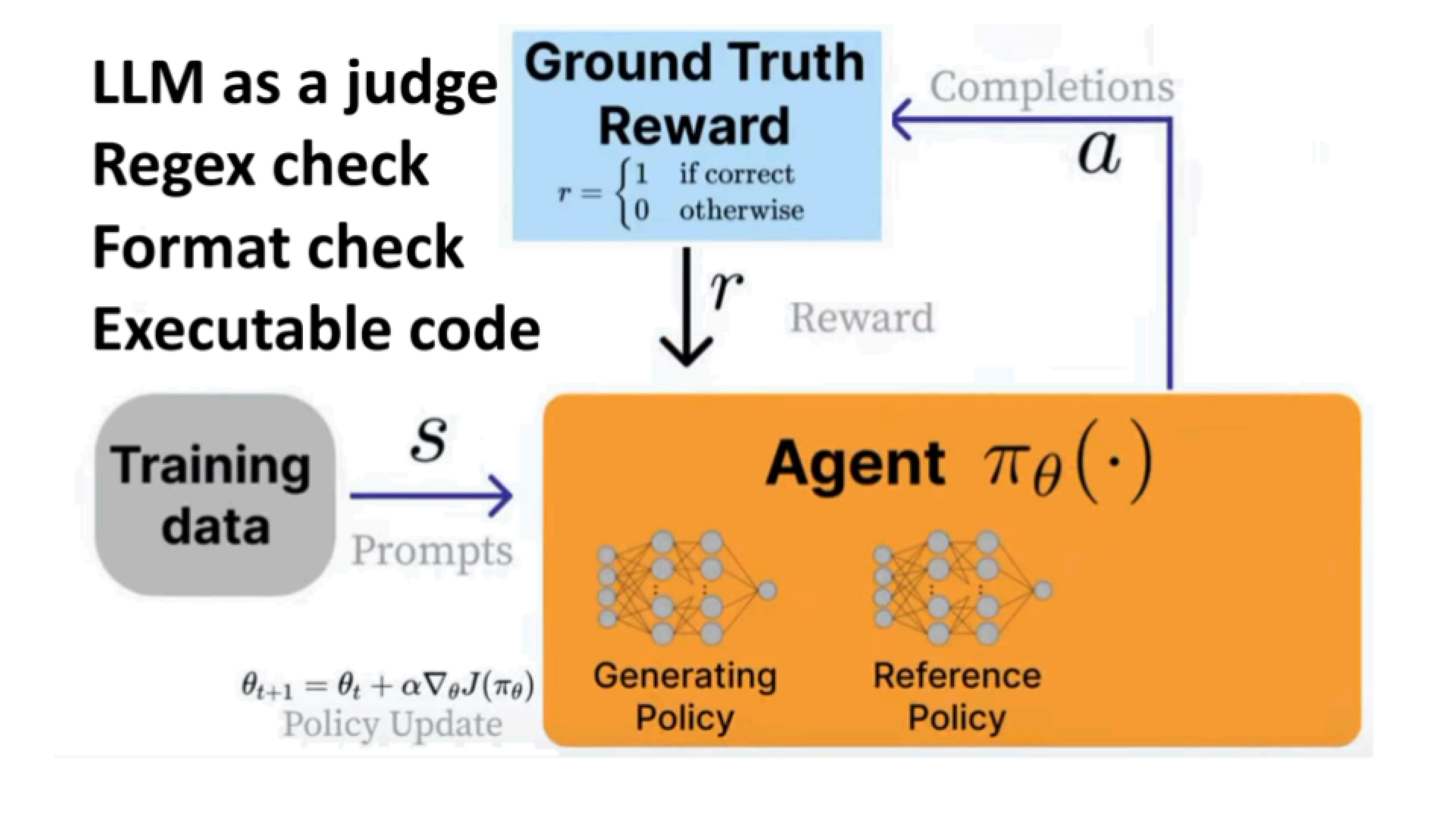


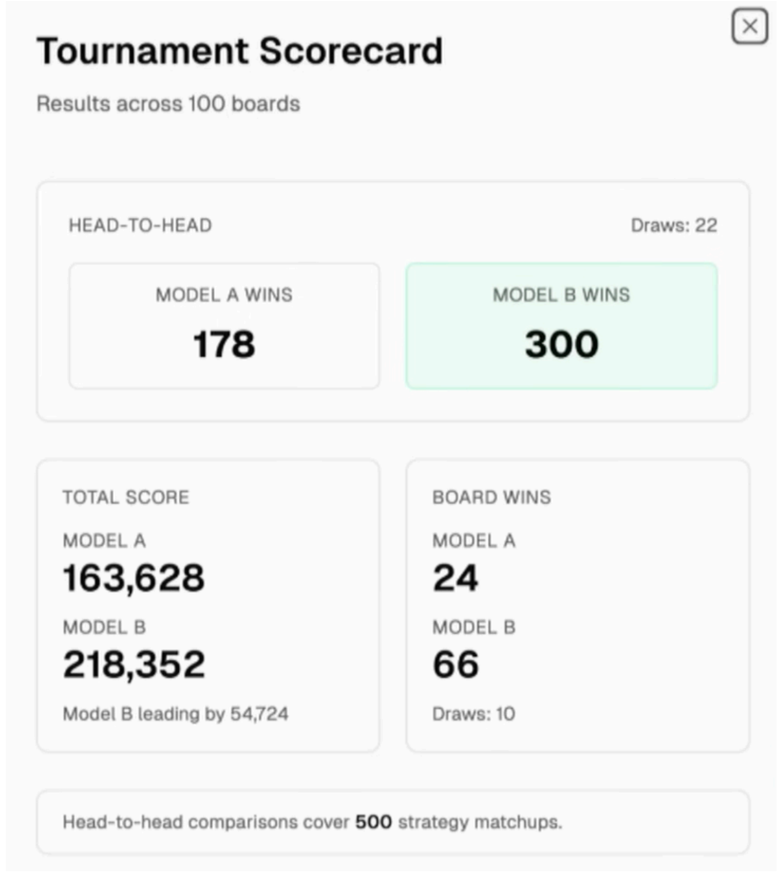


How to maximize “Luck”?

Better Algorithms
Better Reward Functions
Patience :)


GRPO
+
RLVR
Ground Truth Reward
$r = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$
Completions
a
r
Reward
Training data
s
Prompts
Agent $\pi_\theta(\cdot)$
Generating Policy
Reference Policy
$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\pi_\theta)$
Policy Update


LLM as a judge
Regex check
Format check
Executable code
Ground Truth Reward
$r = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$
Completions
a
r
Reward
Training data
S
Prompts
Agent $\pi_\theta(\cdot)$
Generating Policy
Reference Policy
$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\pi_\theta)$
Policy Update

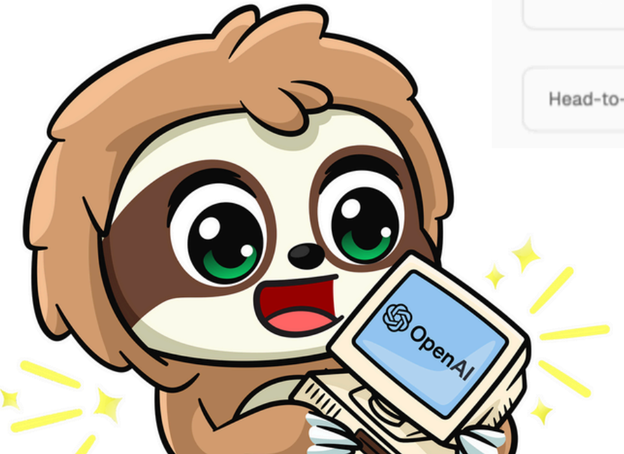


gpt-oss

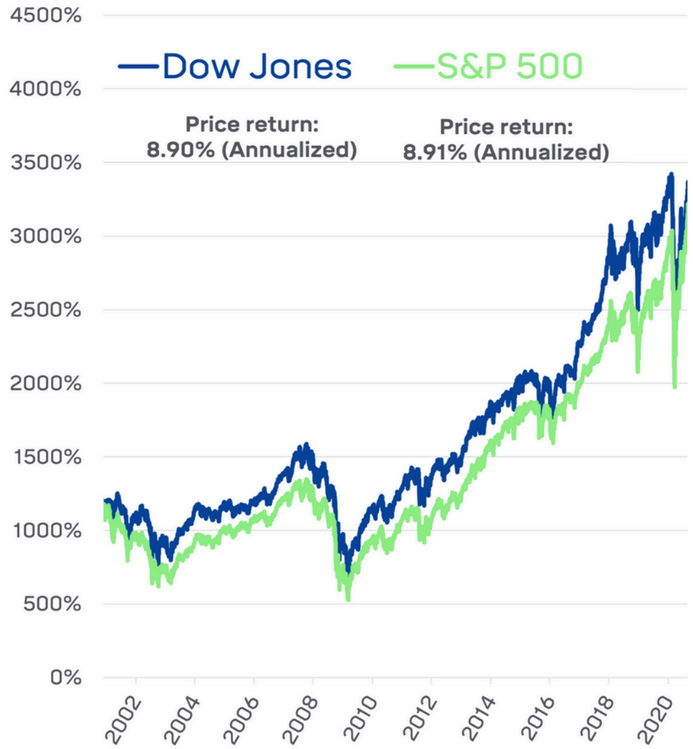
gpt-oss trained with RL consistently outperforms on 2048

Goal: Make gpt-oss play the 2048 game with GRPO

Model auto creates strategies:
Reward if wins
Penalize if loses



unsloth



✅ Profit Gain
❌ Loss

✅ Gets 2048
❌ Infinite loop
❌ Failed executing
☀️ Gets 1024

$$\int \frac{1}{\sqrt{1-x^2}}\,dx = \sin^{-1}x + C$$

$$\int \frac{1}{1+x^2}\,dx = \tan^{-1}x + C$$

$$\int \frac{1}{|x|\sqrt{x^2-1}}\,dx = \sec^{-1}x + C$$

$$\int \sin^n(x)dx = \frac{-1}{n}\sin^{n-1}(x)\cos(x) + \frac{n-1}{n}\int \sin^{n-2}(x)dx$$

$$\int \cos^n(x)dx = \frac{1}{n}\cos^{n-1}(x)\sin(x) + \frac{n-1}{n}\int \cos^{n-2}(x)dx$$

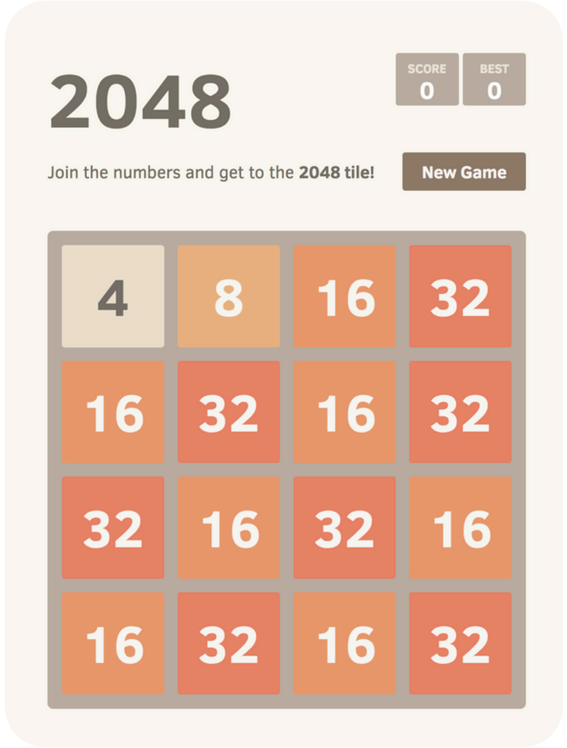
$$\int \tan^n(x)dx = \frac{1}{n-1}\tan^{n-1}(x) - \int \tan^{n-2}(x)dx$$

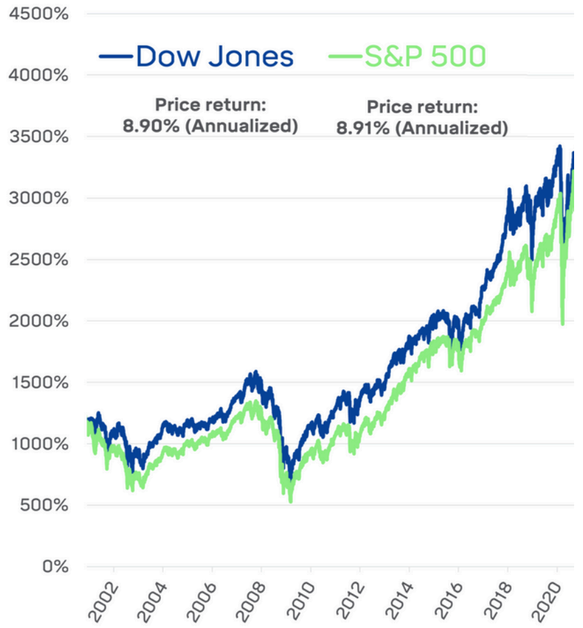
$$\int \sec^n(x)dx = \frac{1}{n-1}\sec^{n-2}(x)\tan(x) + \frac{n-2}{n-1}\int \sec^{n-2}(x)$$

$$\int \csc^n(x)dx = \frac{-1}{n-1}\csc^{n-2}(x)\cot(x) + \frac{n-2}{n-1}\int \csc^{n-2}(x)d$$

✅ Correct answer
❌ Incorrect answer
☀️ Symbol search

unsloth





$$\int \frac{1}{\sqrt{1-x^2}}\,dx = \sin^{-1}x + C$$

$$\int \frac{1}{1+x^2}\,dx = \tan^{-1}x + C$$

$$\int \frac{1}{|x|\sqrt{x^2-1}}\,dx = \sec^{-1}x + C$$

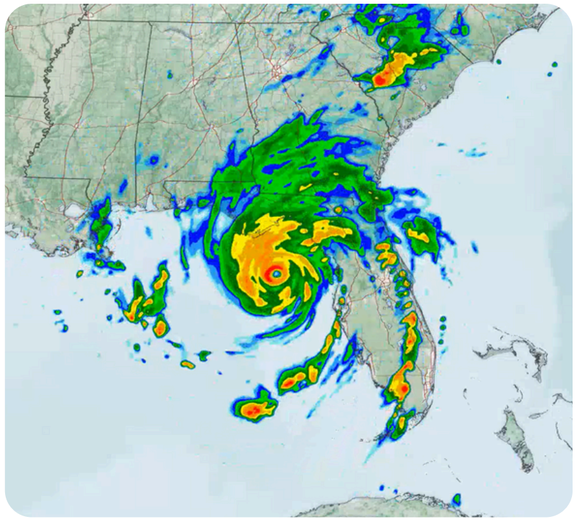
$$\int \sin^n(x)dx = \frac{-1}{n}\sin^{n-1}(x)\cos(x) + \frac{n-1}{n}\int \sin^{n-2}(x)dx$$

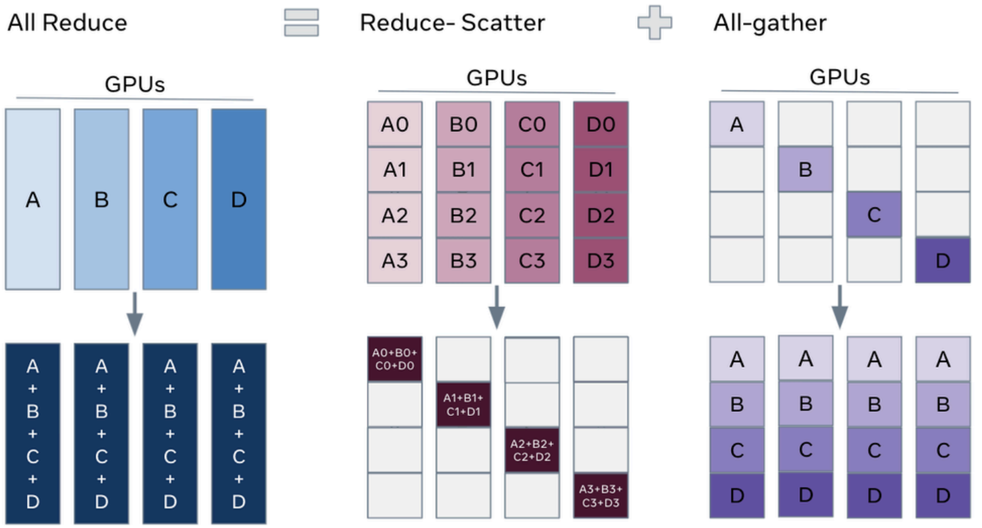
$$\int \cos^n(x)dx = \frac{1}{n}\cos^{n-1}(x)\sin(x) + \frac{n-1}{n}\int \cos^{n-2}(x)dx$$

$$\int \tan^n(x)dx = \frac{1}{n-1}\tan^{n-1}(x) - \int \tan^{n-2}(x)dx$$

$$\int \sec^n(x)dx = \frac{1}{n-1}\sec^{n-2}(x)\tan(x) + \frac{n-2}{n-1}\int \sec^{n-2}(x)$$

$$\int \csc^n(x)dx = \frac{-1}{n-1}\csc^{n-2}(x)\cot(x) + \frac{n-2}{n-1}\int \csc^{n-2}(x)d$$





and more!

unsloth

GPT-OSS_(20B)-Reinforcement-Learning-GRPO.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

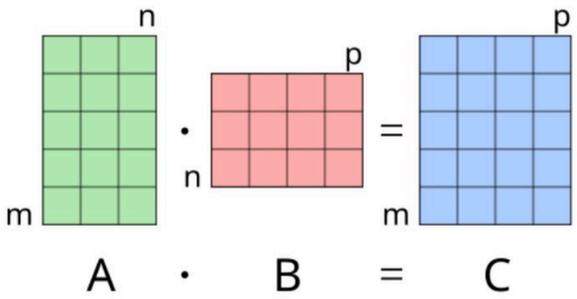
Commands + Code + Text Run all Copy to Drive Connect T4

To run this, press "*Runtime*" and press "*Run all*" on a **free** Tesla T4 Google Colab instance!

OpenAI gpt-oss unsloth

Goal: Make faster kernels with Reinforcement Learning

Our goal is to make a faster matrix multiplication kernel by doing RL on GTP-OSS 20B with Unsloth.



You will learn how to:

1. Counteract **reward hacking** like cheating, caching, laziness.
2. Timing and correctness of kernels and time limits.
3. Making good **reward functions**
4. How to seriously do RL to make optimized CUDA kernels

Optimized matrix multiplication

Numpy has optimized matrix multiplication kernels for CPUs via BLAS optimized operations. For GPUs, one can use CUDA accelerate cuBLAS kernels which PyTorch calls under the hood.

To generate some random matrices to do matrix multiplication, we can do the below:

[ ]

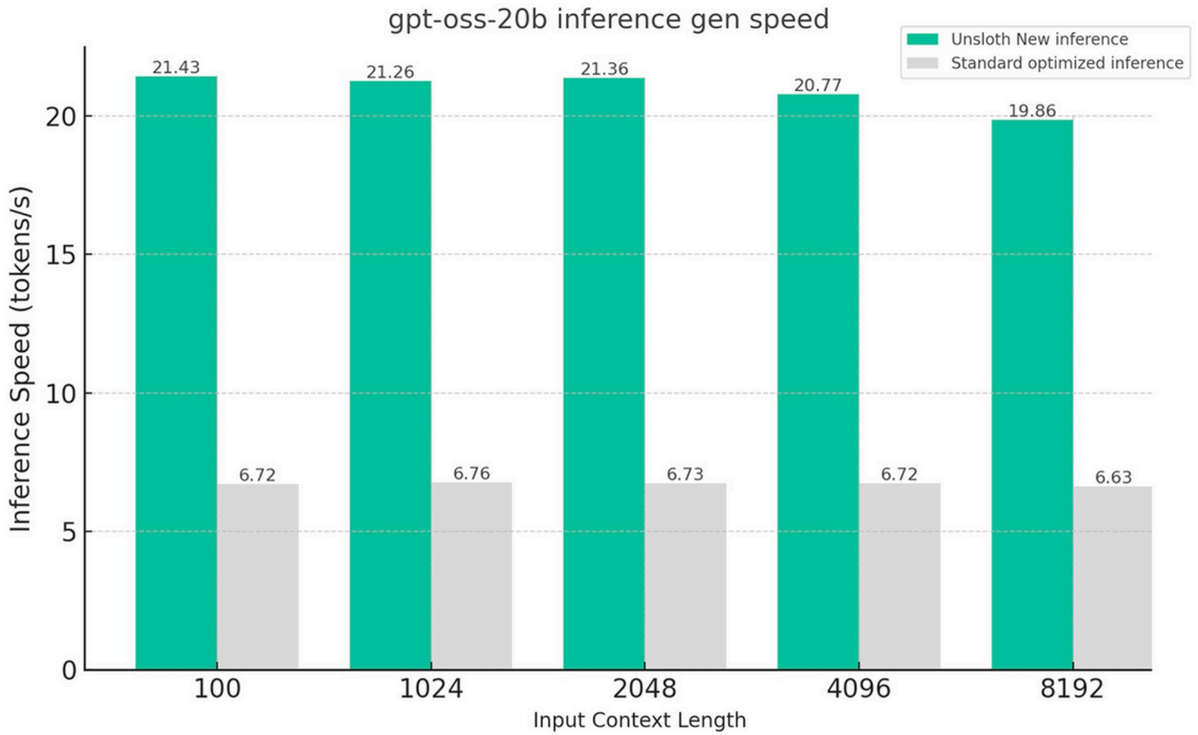
```
import numpy as np
def generate_random_matrices(seed = 3407, n = 256):
```

Intelligence Explosion Automating AI Research

Faster inference == Faster!

gpt-oss

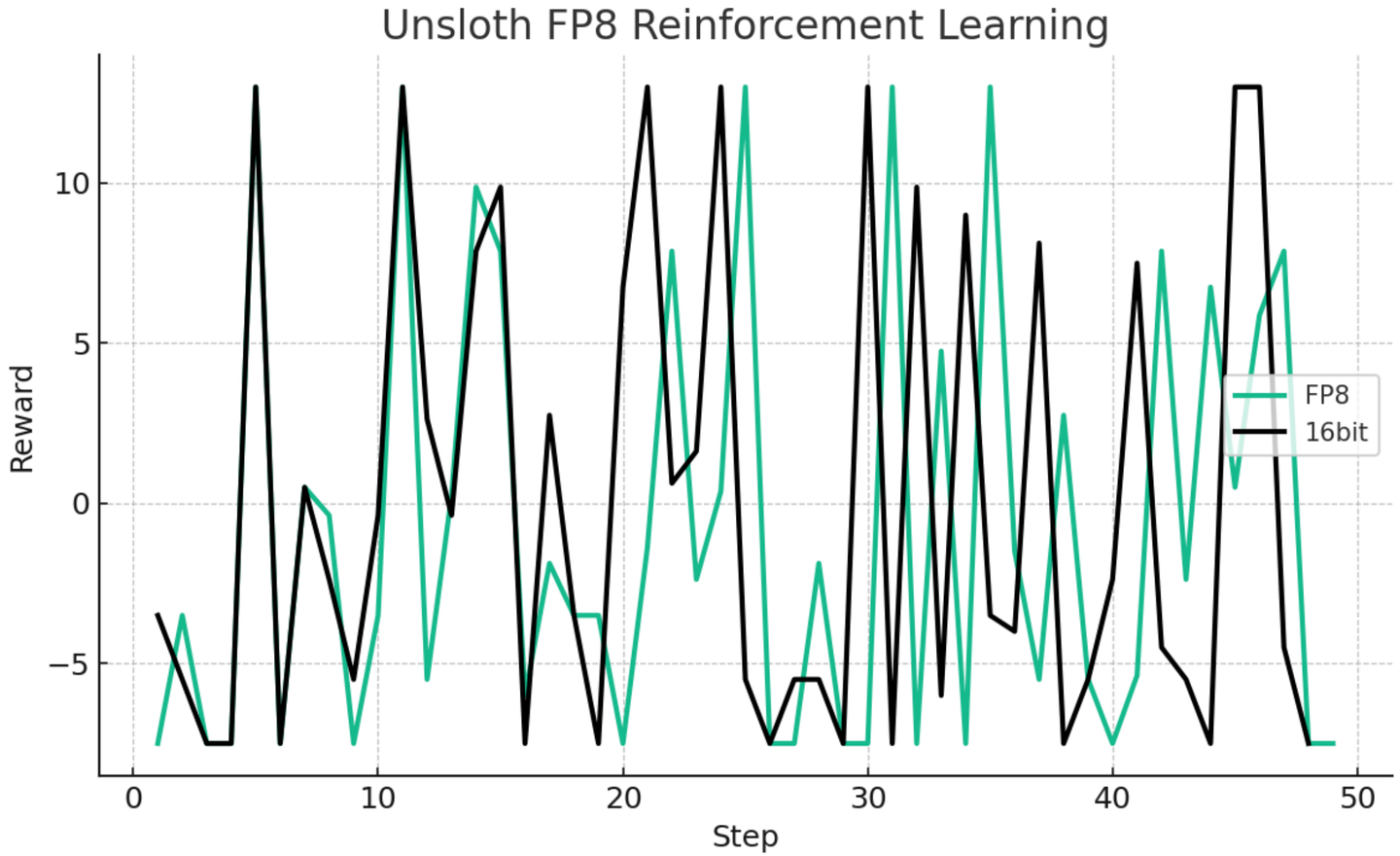
Fastest inference for Reinforcement Learning



BF16 or 4-bit GRPO training - No accuracy degradation

PyTorch TorchAO unsloth

FP8 Reinforcement Learning



1.33x faster
-50% VRAM

“Sucking supervision bits through a straw”

Andrej Karpathy, Dwarkesh Podcast

unsloth

What is the integral of f(x)

By using ...
I can see ...
Wait maybe that's ...
Ok wait let's backtrack ...
Hm I think I got it ...
The answer is g(x)

unsloth

What is the integral of f(x)

By using ...
I can see ...
Wait maybe that's ...
Ok wait let's backtrack ...
Hm I think I got it ...
The answer is g(x)

LLM Judge

unsloth

What is the integral of f(x)

By using ...
I can see ...
Wait maybe that's ...
Ok wait let's backtrack ...
Hm I think I got it ...
The answer is g(x)

Expert help

unsloth

The Art of Scaling Reinforcement Learning Compute for LLMs

Devvrit Khatri[2,*,†], **Lovish Madaan**[1,3,*],
Rishabh Tiwari[4†], **Rachit Bansal**[5†], **Sai Surya Duvvuri**[2†], **Manzil Zaheer**[1†],
Inderjit S. Dhillon[2], **David Brandfonbrener**[1], **Rishabh Agarwal**[6,†]

[1]Meta, [2]UT Austin, [3]UCL, [4]UC Berkeley, [5]Harvard University, [6]Periodic Labs
[*]Equal contribution, [†]Work done at Meta

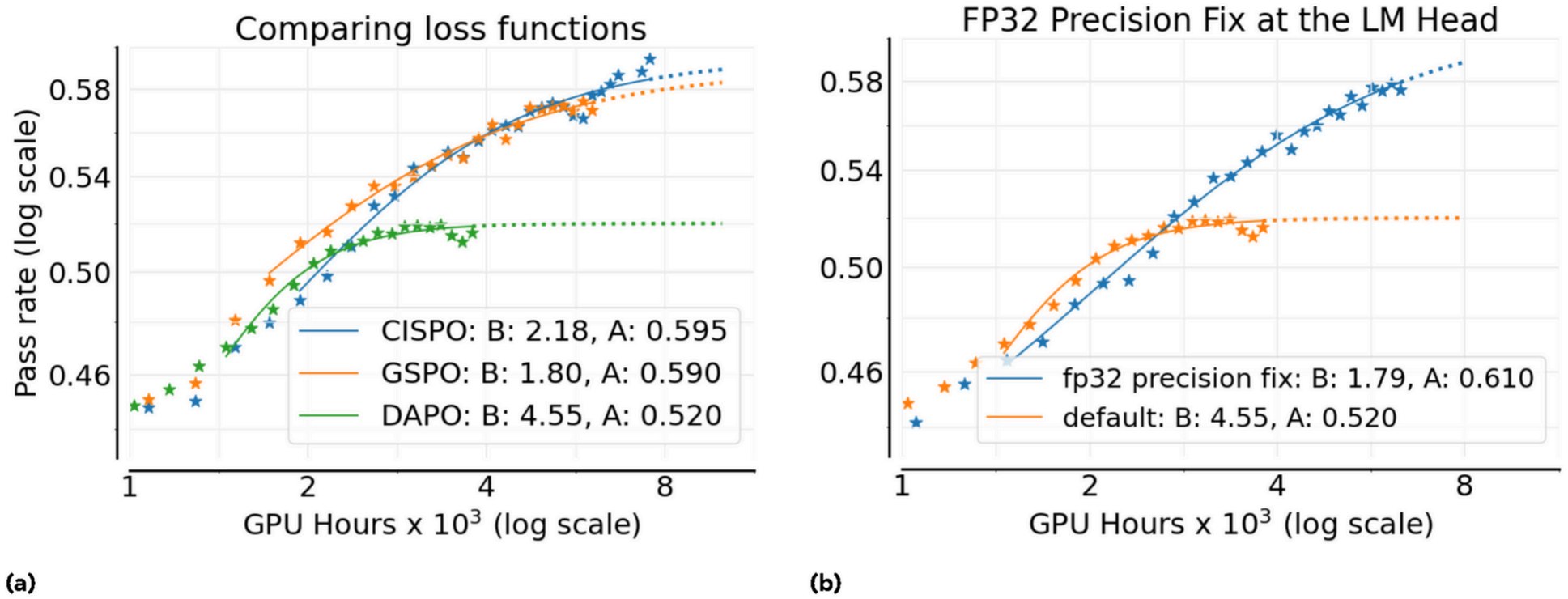
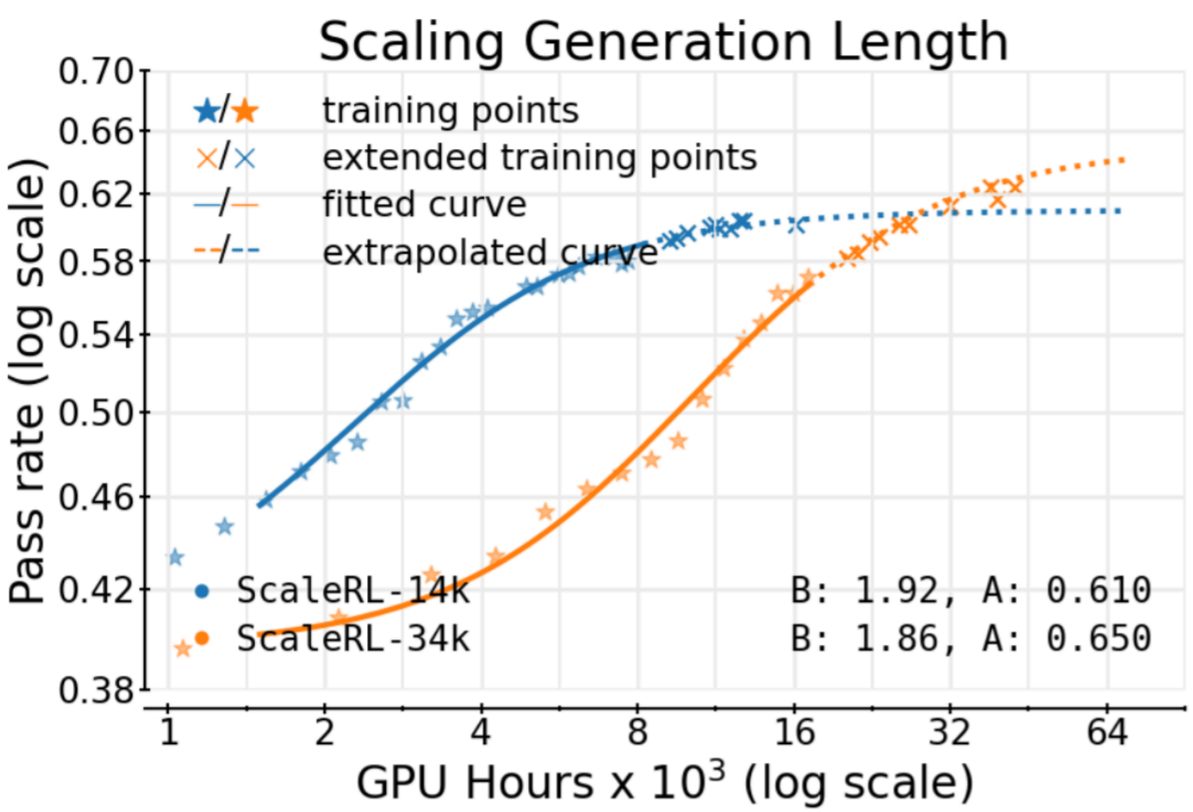
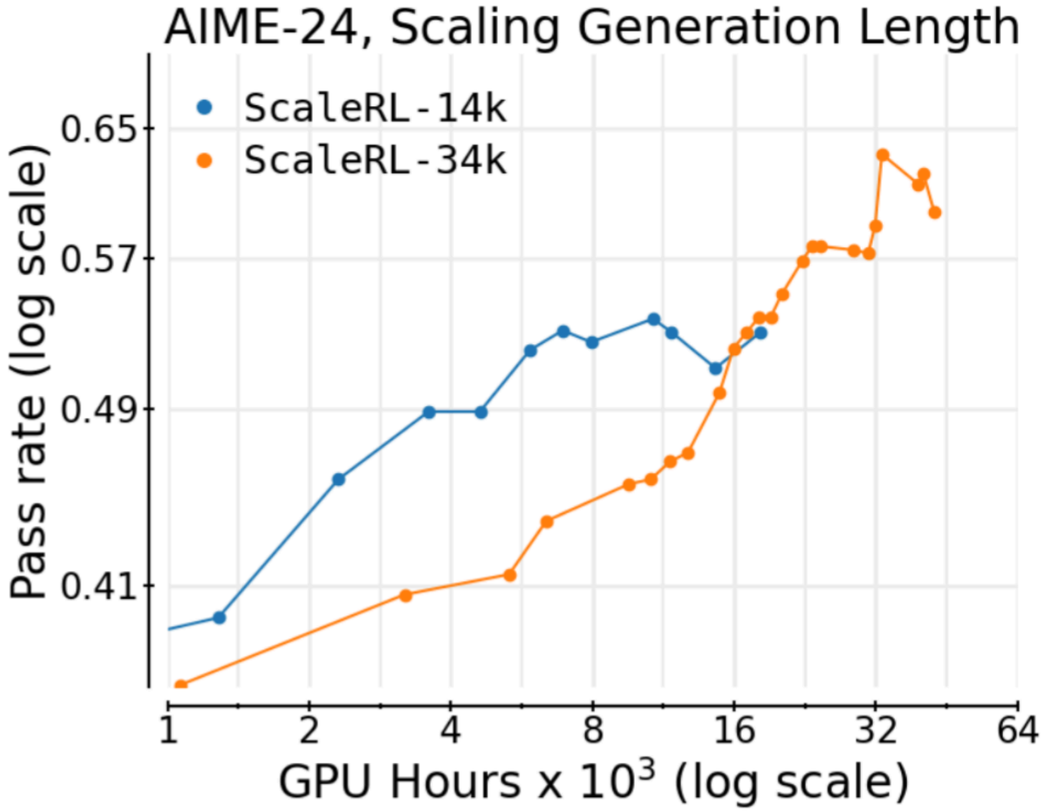


Figure 5 (a) **Comparing popular loss functions**: DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025a), and CISPO (MiniMax et al., 2025). We find CISPO/GSPO achieve a higher asymptotic reward compared to DAPO. (b) **Using FP32 precision** in the final layer (LM head) gives a considerable boost in the asymptotic reward.

Long context wins



(a)

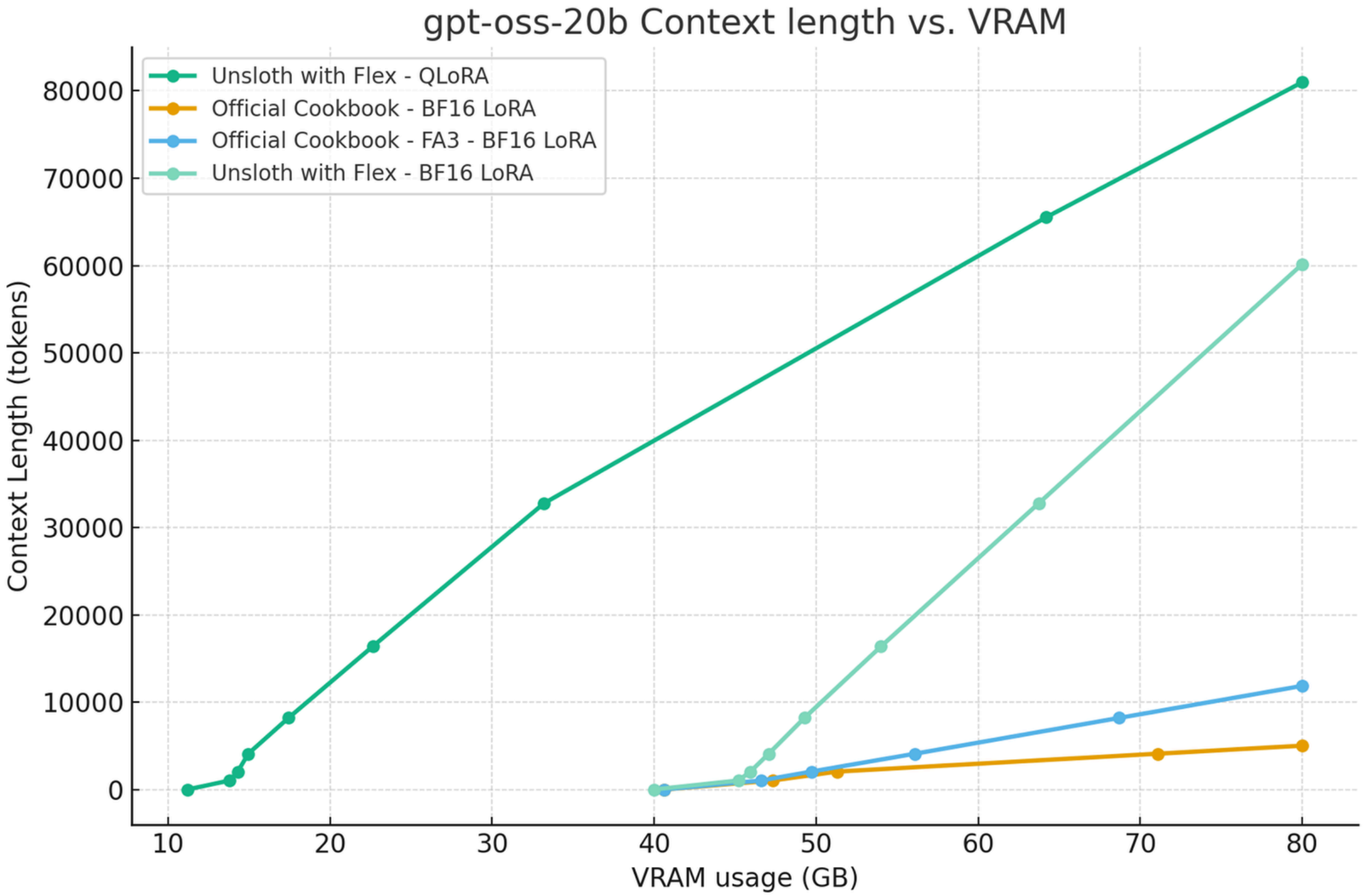


(b)

Figure 9 Scaling RL Generation Length. While long-context RL is less efficient initially, it eventually surpasses the performance of the smaller-context run. This trend is observed on both the *iid* validation set (left) as well as downstream evaluations (right).

gpt-oss

ultra long context training



Large batches win

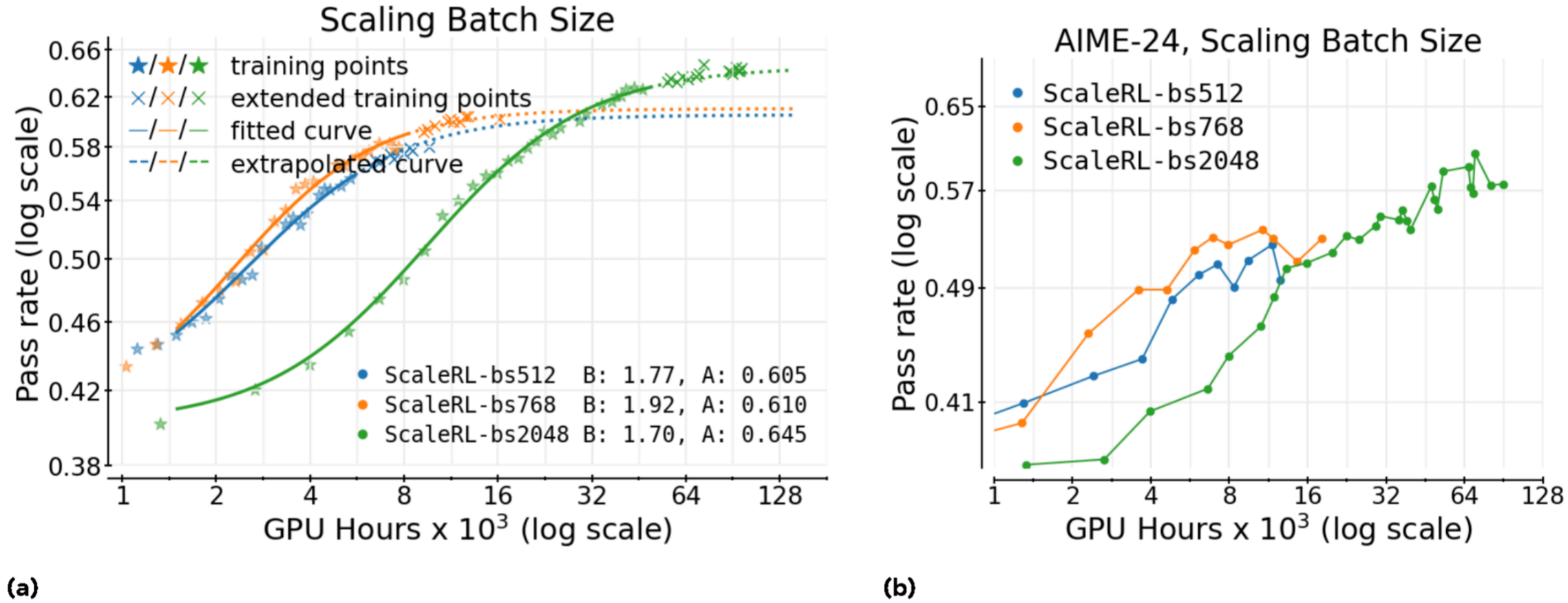
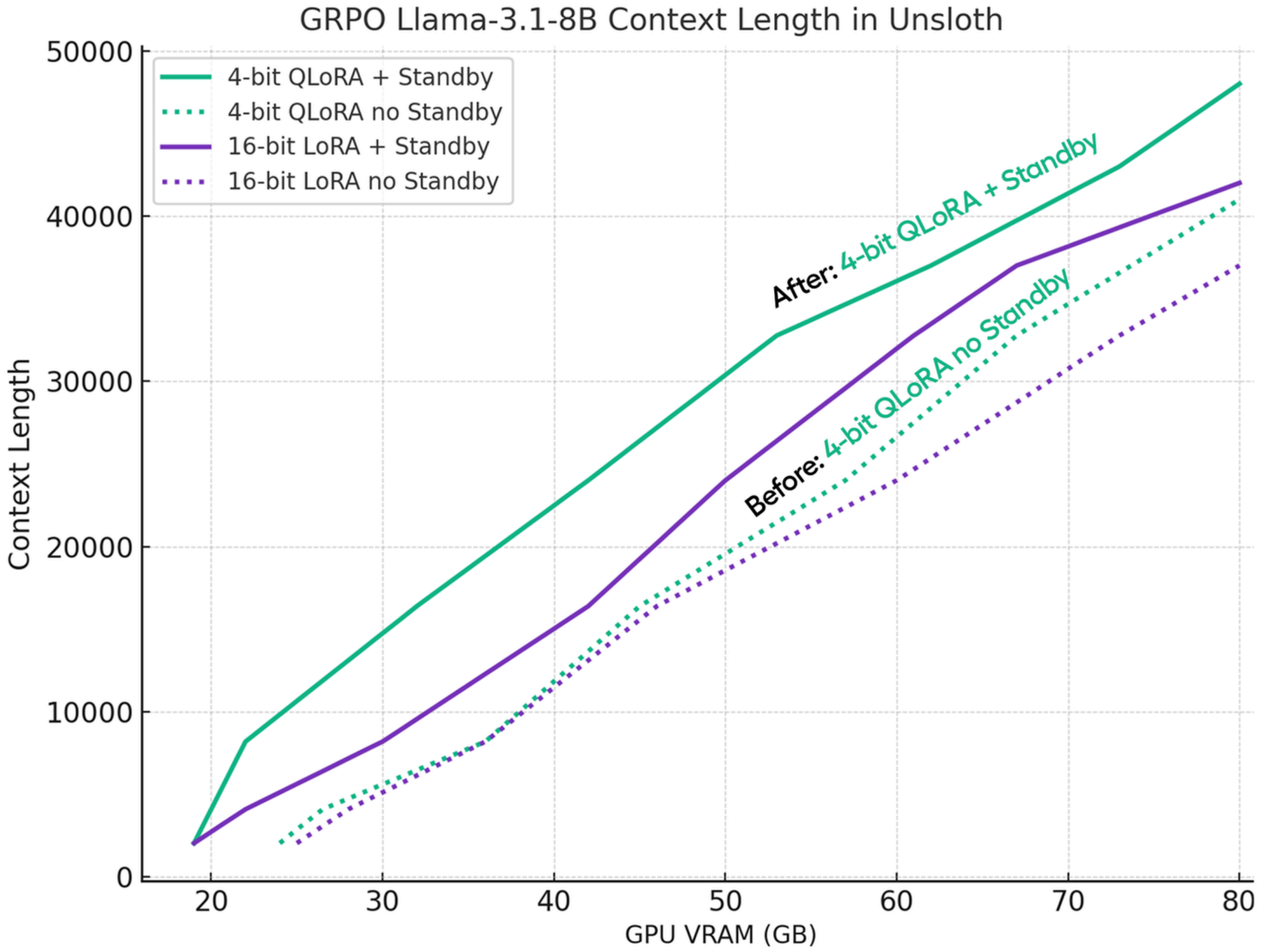


Figure 10 Scaling RL batch size. larger batch size is slower in training but settles at a higher asymptote. Batch size show an inverse trend initially where smaller values seem better at lower compute budget, but reach a higher asymptotic performance at larger scale.

Memory Efficient RL

50% less VRAM • 10x context • 10% faster • Same accuracy



LoRA Without Regret

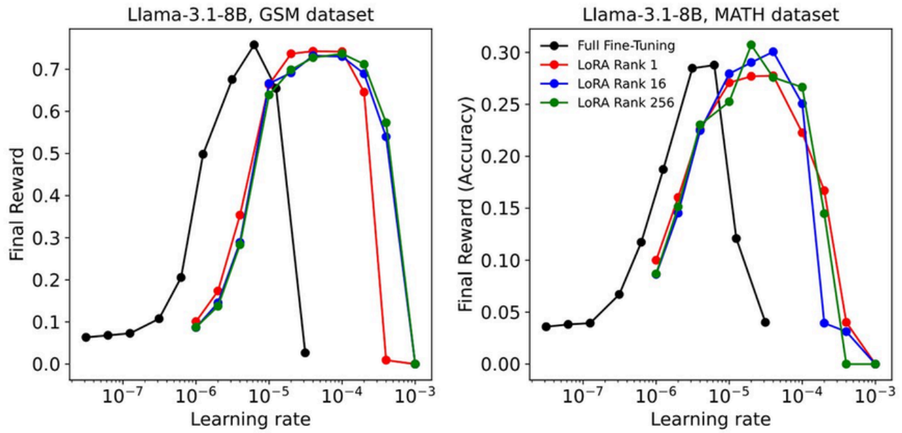
John Schulman in collaboration with others at Thinking Machines

Sep 29, 2025

Introduction
What matters for LoRA
Methods and results
LoRA rank
Batch size effects
Layers Where LoRA Is Applied
Reinforcement learning
Setting LoRA hyperparameters
Optimal learning rate and rank
Parametrization invariances
Optimal learning rates for LoRA vs. FullFT

Today's leading language models contain upwards of a trillion parameters, pretrained on tens of trillions of tokens. Base model performance keeps improving with scale, as these trillions are necessary for learning and representing all the patterns in written-down human knowledge.

In contrast, post-training involves smaller datasets and generally focuses on narrower domains of knowledge and ranges of behavior. It seems wasteful to use a terabit of weights to represent updates from a gigabit or megabit of training data. This intuition has motivated parameter efficient fine-tuning (PEFT), which adjusts a large network by updating a much smaller set of parameters.



What matters for LoRA

This article covers a series of supervised fine-tuning and reinforcement learning experiments we conducted to determine the conditions under which LoRA matches FullFT efficiency. To this end, we did a few things differently from previous experiments on LoRA

We find that:

- LoRA performs equivalently to FullFT for reinforcement learning even with small ranks. We find that RL requires very low capacity, a result we anticipated based on information-theoretical arguments.

THINKING MACHINES

Unsloth's LoRA Hyperparameter Guide recommends using higher values of α for high-rank LoRA, e.g. by avoiding the $1/r$ scaling. This is also equivalent to increasing init_A/LR_A. When we increase α, LR_A and LR_B need to be lowered in compensation to get the same update size. This in turn simply makes LR_A smaller relative to init_A.

🧠 LoRA Hyperparameters Guide

Optimal lora rank. alpha, number of epochs, batch size & gradient accumulation, QLoRA vs LoRA, target modules and more!

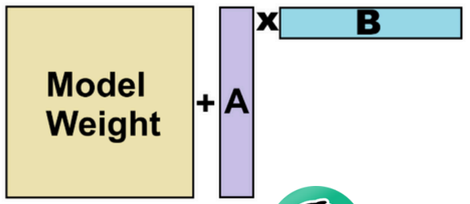
LoRA hyperparameters are adjustable parameters that control how Low-Rank Adaptation (LoRA) fine-tunes LLMs. With many options (such as learning rate and epochs) and millions of possible combinations, selecting the right values is crucial for achieving accuracy, stability, quality, and fewer hallucinations during fine-tuning.

You'll learn the best practices for these parameters, based on insights from hundreds of research papers and experiments, and see how they impact the model. **While we recommend using Unsloth's defaults**, understanding these concepts will give you full control.

The goal is to change hyperparameter numbers to increase accuracy while counteracting overfitting or underfitting. Overfitting occurs when the model memorizes the training data, harming its ability to generalize to new, unseen inputs. The objective is a model that generalizes well, not one that simply memorizes.

❓ But what is LoRA?

In LLMs, we have model weights. Llama 70B has 70 billion numbers. Instead of changing all 70b numbers, we instead add thin matrices A and B to each weight, and optimize those. This means we only optimize 1% of weights.



Instead of optimizing Model Weights [illegible] optimize 2 tiny matrices A an[illegible]

unsloth

What is 2+2?

%!%%^@
ASDSD
ASD123
9876668
askdkasd
The answer is 4.

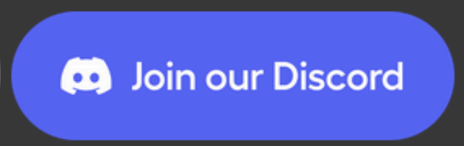
Reward hacking

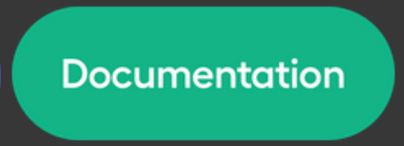
unsloth

OpenEnv: Agentic Execution Environments

We're using the new OpenEnv library which has over 2000+ environments for RL!

To run this, press "*Runtime*" and press "*Run all*" on a **free** Tesla T4 Google Colab instance!

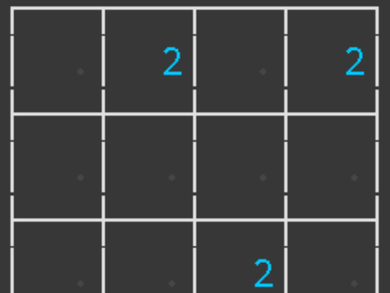




Join Discord if you need help + ⭐ *Star us on Github* ⭐

To install Unsloth your local device, follow our guide.

```
action = OpenSpielAction(action_id = 0, game_name = "2048")
result = openenv_process.step(action)
current_state = result.observation
print(render_board(current_state))
```




```
def strategy(board):
    size = len(board)
    # helper to detect if move merges
    def can_move_dir(dx,dy):
        for i in range(size):
            for j in range(size):
```

Patience is all you need

RL is not “dumb”

Unsloth AI Team Company ✓ Verified

https://unsloth.ai UnslothAI unslothai

Hugging Face

Activity Feed Follow 10,030

AI & ML interests

Open Source AI 💚

Dynamic 2.0 quants

gpt-oss, Qwen3, Mistral, Gemma 3

Expect Bug fixes + updates for chat templates, tokenizers etc.

GGUF, 4-bit, original

Collections 21

Collapse

Unsloth Dynamic 2.0 Quants

New 2.0 version of our Dynamic GGUF + Quants. Dynamic 2.0 …

- unsloth/DeepSeek-V3.1-GGUF — 671B • Updated 9 days ago • 45.9k • 82
- unsloth/Qwen3-Coder-30B-A3B-Instruct-GGUF — Text Generation • 31B • Updated A… • 171k • 227
- unsloth/Qwen3-30B-A3B-Instruct-2507-GGUF — 31B • Updated Jul 31 • 74.3k • 220
- unsloth/Qwen3-30B-A3B-Thinking-2507-GGUF

gpt-oss

OpenAI's gpt-oss-20b and gpt-oss-120b is here! The powerful …

- unsloth/gpt-oss-20b-GGUF — Text Generation • 21B • Updated 21… • 363k • 374
- unsloth/gpt-oss-120b-GGUF — Text Generation • 117B • Updated 1… • 128k • 140
- unsloth/gpt-oss-20b-unsloth-bnb-4bit — Text Generation • 21B • Updated Aug 8 • 163k • 23
- unsloth/gpt-oss-120b-unsloth-bnb-4bit

Qwen3

Qwen's new Qwen3 models. In Unsloth Dynamic 2.0, GGUF, 4-…

- unsloth/Qwen3-30B-A3B-Instruct-2507-GGUF — 31B • Updated Jul 31 • 74.3k • 220
- unsloth/Qwen3-4B-Instruct-2507-GGUF — 4B • Updated 22 days ago • 63k • 60
- unsloth/Qwen3-4B-Thinking-2507-GGUF — 4B • Updated Aug 6 • 34.2k • 45

DeepSeek-V3.1

DeepSeek's new 3.1 update to their V3 models!

- unsloth/DeepSeek-V3.1-GGUF — 671B • Updated 9 days ago • 45.9k • 82
- unsloth/DeepSeek-V3.1 — Text Generation • 685B • Updated 21 d… • 156 • 4
- unsloth/DeepSeek-V3.1-BF16 — Text Generation • 684B • Updated 21 days ago • 993

Thank you!

⭐ Star us on GitHub