Lecture Series: Reinforcement Learning of Large Language Models

Ernest K. Ryu

University of California, Los Angeles

Spring 2025

Course plan

Prerequisite: Students are expected to have basic familiarity with deep learning, at the level of image classification. No prior experience with reinforcement learning (RL) or large language models (LLMs) is assumed. For the deep RL lectures, students should be familiar with conditional expectations and the tower property (law of total expectation).

Course plan:

- Prologue
- Chapter 1: Deep Reinforcement learning.
    - From definitions of MDPs to PPO and GRPO type algorithms.
- Chapter 2: Large Language Models.
    - From basic notions of NLP to modern transformer-based language models.
- Chapter 3: Reinforcement Learning of Large Language Models.
    - RLHF and DeepSeek-R1-style RLVR.

Prologue:
Summer of RL and AI

Ernest K. Ryu

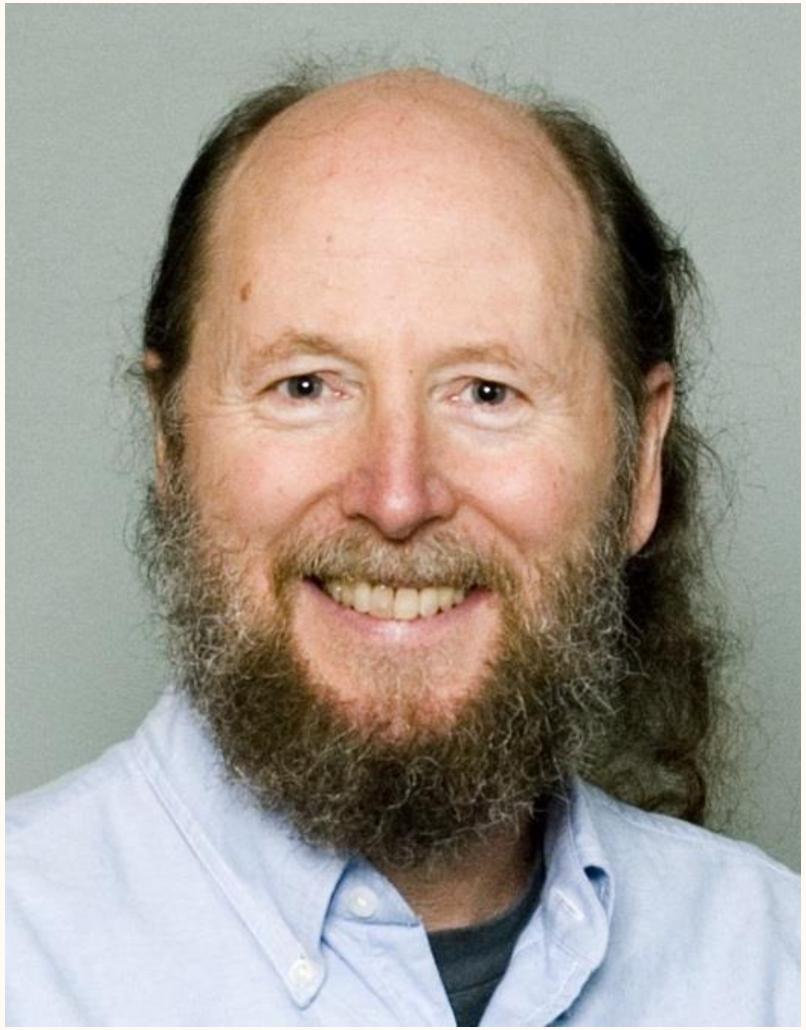
University of California, Los Angeles

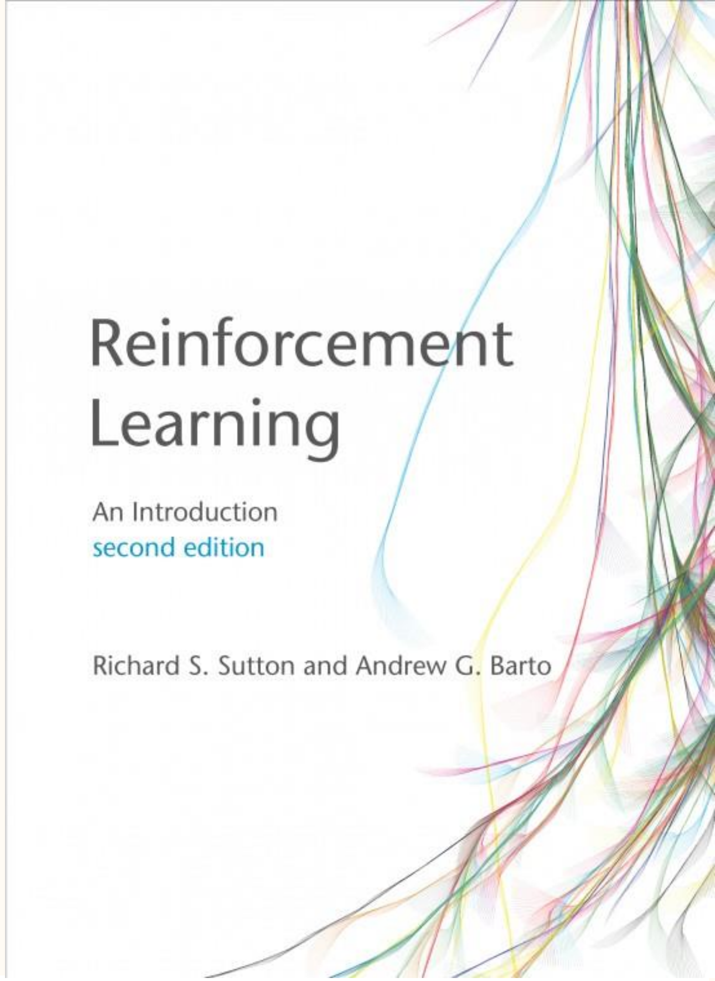
Richard M. Sutton

One of the founding fathers of reinforcement learning. Turing awardee.

In his essay titled “The Bitter Lesson”, Sutton articulates a crucial principle in modern deep learning. A must-read for anybody working in AI.

Let’s read it together.





The Bitter Lesson

“The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent."

The Bitter Lesson

“In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that “brute force” search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

A similar pattern of research progress was seen in computer Go, … **Search** and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing **search** and learning.”

The Bitter Lesson

“In speech recognition, … computer vision, there has been a similar pattern.

This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.”

The Bitter Lesson

“One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are **search** and learning.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds… instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.”

— Richard M. Sutton —

March 13, 2019

How to train a large language model (LLM)

Step 1. Get internet-scale text data. All books, all writings ever written by humans.

- 2017 Original Transformer – 100 Million Tokens (10 Library shelves)
- 2018 GPT 1 – 600 Million Tokens (60 shelves)
- 2019 GPT 2 – 28 Billion Tokens (2800 shelves)
- 2020 GPT 3 – 300 Billion Tokens (30,000 shelves)
- 2022 PALM – 780 Million Tokens (78,000 shelves)
- 2023 GPT4 – 1.3 Trillion Tokens
  (130,000 shelves = 650 km of shelves side by side)

1 library bookshelf = 10 million tokens

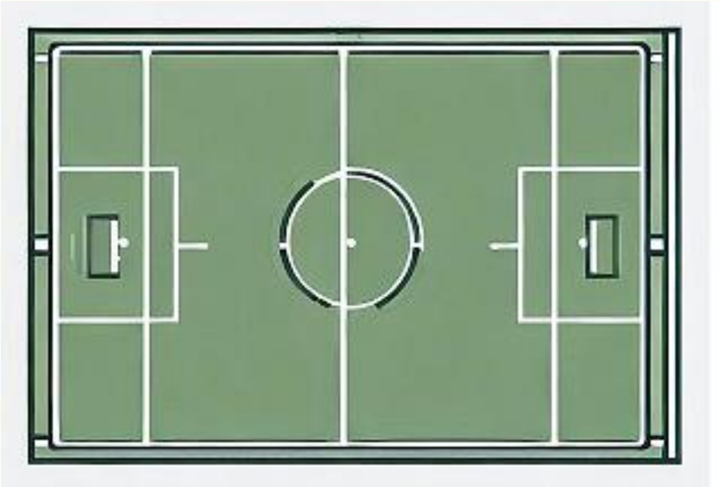
How to train a large language model (LLM)

Step 2. Create a large transformer architecture.

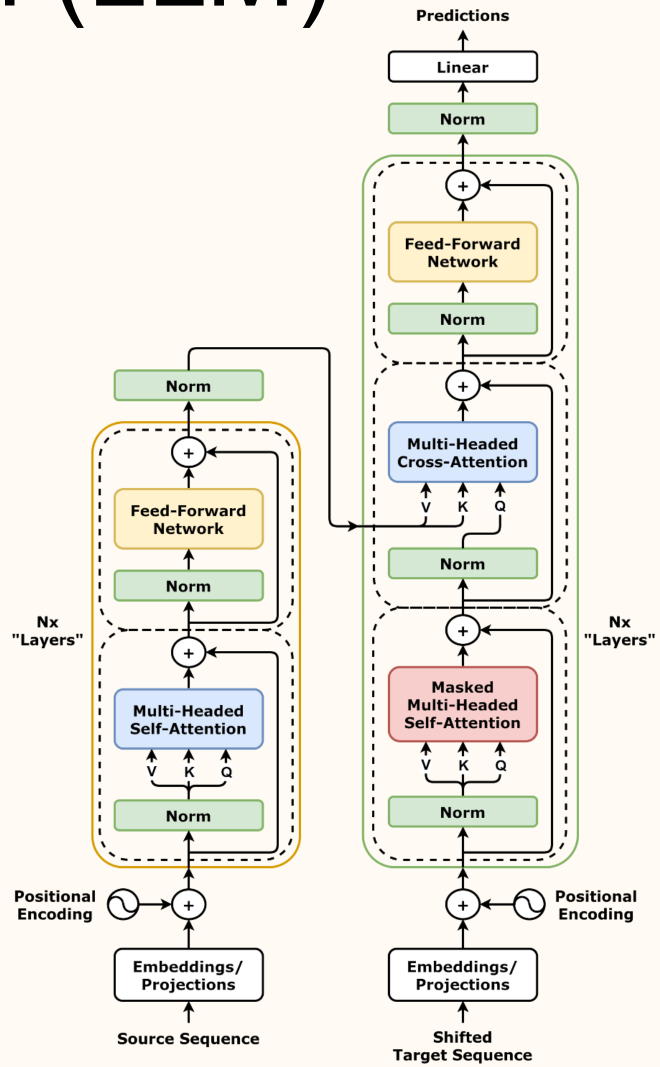
Transformers has many parameters. Imagine the parameters written into Excel sheets. Let’s visualize the size.

	A	B	C	D	E	F	G	H
1	0.938	0.395	0.4	0.498	0.408	0.654	0.153	0.961
2	0.859	0.508	0.564	0.2	0.392	0.982	0.571	0.114
3	0.206	0.567	0.394	0.52	0.195	0.193	0.842	0.747
4	0.366	0.157	0.77	0.3	0.003	0.429	0.395	0.261
5	0.872	0.796	0.709	0.37	0.62	0.36	0.029	0.531
6	0.13	0.446	0.796	0.786	0.262	0.226	0.062	0.57
7	0.5	0.288	0.073	0.555	0.43	0.205	0.052	0.937
8	0.265	0.864	0.539	0.426	0.688	0.833	0.388	0.339
9	0.137	0.04	0.641	0.353	0.48	0.499	0.888	0.919
10	0.008	0.607	0.685	0.467	0.138	0.188	0.388	0.697
11	0.58	0.038	0.819	0.094	0.419	0.014	0.723	0.351
12	0.121	0.314	0.098	0.052	0.581	0.085	0.803	0.618
13	0.369	0.588	0.486	0.518	0.49	0.966	0.091	0.292
14	0.225	0.978	0.49	0.03	0.427	0.949	0.387	0.453
15	0.855	0.048	0.227	0.204	0.931	0.878	0.865	0.91
16	0.324	0.856	0.751	0.656	0.444	0.42	0.175	0.438

Assume each cell size of 1cm x 1cm



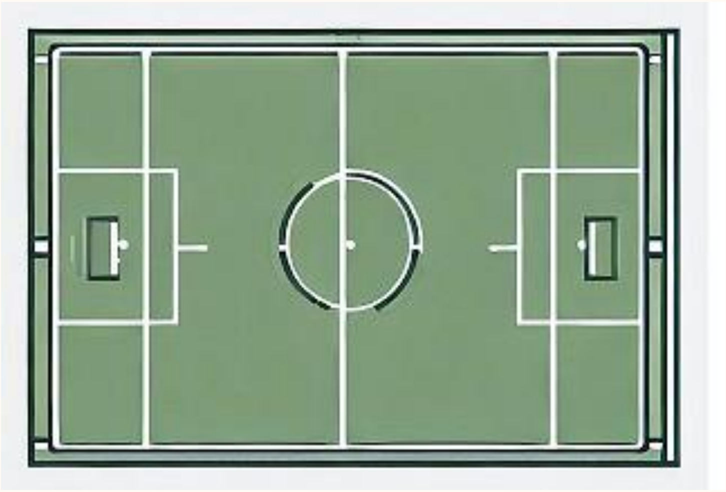
1 Football field = 60 million parameters



How to train a large language model (LLM)

Step 2. Create a large transformer architecture.

- 2017 Original Transformer – 65 Million Parameters (1 Football field)
- 2018 GPT 1 – 117 Million Parameters (2 fields)
- 2019 GPT 2 – 1500 Million Parameters (20 fields)
- 2020 GPT 3 – 175,000 Million Parameters (2500 fields)
- 2022 PALM – 540,000 Million Parameters (7700 fields)
- 2023 GPT4 – 1.8 Trillion parameters
  (30,000 fields = 180 km^2)
  (Washington DC = 177 km^2)

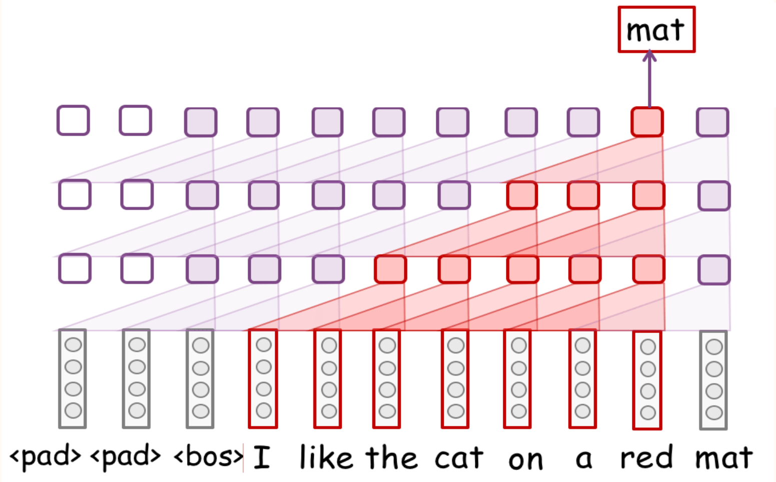


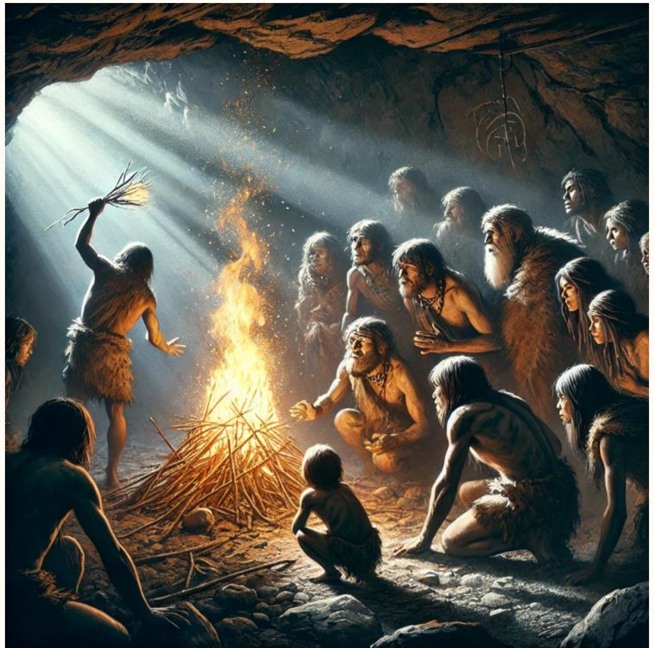
1 Football field = 60 million parameters

How to train a large language model (LLM)

Step 3. Train with next token prediction. Provide a partial sentence to the language model and have it predict the next word.

- 2017 Original Transformer – 4×10^{17}FLOPs. (45 days training time on a PC)
- 2018 GPT 1 – 4×10^{19}FLOPs. (13 years on a PC)
- 2019 GPT 2 – 5×10^{21}FLOPs. (1600 years on a PC)
- 2020 GPT 3 – 3×10^{23}FLOPs. (100,000 years on a PC)
- 2022 PALM – 3×10^{24}FLOPs. (800,000 years on a PC)
- 2023 GPT4 – 2×10^{25}FLOPs. (7,000,000 years on a PC) (Discovery of fire $\approx$ 2,000,000 years ago)





LLMs work really well

ChatGPT o3-mini-high

Consider the ODE y'=f(y,t), where f is sufficiently smooth. I am performing an implicit Euler update with stepsize h>0. Will there by a solution to the implicit update if h>0 is sufficiently small?

Reasoned for 11 seconds >

The answer is yes. Under the assumption that f is sufficiently smooth—meaning in particular that it is locally Lipschitz—the implicit Euler update

$$y_{n+1} = y_n + h\, f(y_{n+1}, t_{n+1})$$

can be reformulated as finding a fixed point of the mapping

$$\phi(y) = y_n + h\, f(y, t_{n+1}).$$

If f is locally Lipschitz with constant L, then for y and z in a small neighborhood, we have:

$$\|\phi(y) - \phi(z)\| = h\|f(y, t_{n+1}) - f(z, t_{n+1})\| \leq hL\|y - z\|.$$

Thus, if $hL < 1$, the mapping ϕ is a contraction. By the Banach fixed-point theorem, a unique fixed point exists, which is exactly the solution to the implicit Euler update.

Additionally, one can invoke the implicit function theorem here. Since f is smooth, the equation

$$F(y) = y - y_n - h\, f(y, t_{n+1}) = 0$$

has a Jacobian (with respect to y) that is close to the identity matrix when h is small. This guarantees that the derivative is invertible, ensuring the existence (and uniqueness) of a solution in a neighborhood of y_n.

In summary, for a sufficiently small step size h (specifically, when $hL < 1$), there will indeed be a unique solution to the implicit Euler update.

LLMs can reliably solve 1st-year-PhD-level mathematics problems. (Such knowledge are in textbooks and probably in LLM's pretraining data.)

Often better than classical IR-based search (Google), since the query is more robust against typos, different notation, different terminology, and minor variations to the problem.

The answer is not always perfect (although it is perfect in this example), but you can work towards a correct answer or hint by interactively asking follow-up questions or pointing out errors.

LLMs work really well

ChatGPT o3-mini-high

Define $\varphi\colon \mathbb{R}\rightarrow\mathbb{R}$ as
\[
\varphi(s)=
\left\{
\begin{array}{ll}
(x+1)^2&\text{for }x<-1\\

Define $\varphi\colon \mathbb{R}\to\mathbb{R}$ as

$$\varphi(s) = \begin{cases} (x+1)^2 & \text{for } x < -1 \\ 0 & \text{for } -1 \le x \le +1 \\ (x-1)^2 & \text{for } +1 < x. \end{cases}$$

Consider the optimization problem

$$\underset{x,y,z\in\mathbb{R}}{\text{minimize}} \quad -xy - yz - zx + \varphi(x) + \varphi(y) + \varphi(z).$$

Implement Python code that executes alternating minimization, updating x, y, and z sequentially.

}\mbox{minimize}}&
-xy-yz-zx+\varphi(x)+\varphi(y)+\varphi(z)
.
\end{array}
\]
Implement Python code that executes alternating minimization, updating x, y, and z sequentially.

Reasoned for 1m 16s >

Below is one complete Python implementation. In this code we first define the function

$$\varphi(x) = \begin{cases} (x+1)^2, & x < -1, \\ 0, & -1 \le x \le 1, \\ (x-1)^2, & x > 1, \end{cases}$$

and then note that when updating (say) x with y, z fixed the minimization problem becomes

$$\min_{x\in\mathbb{R}} \varphi(x) - x\,(y+z).$$

Because φ is piecewise quadratic (or linear on $[-1,1]$) one can solve each case in closed form. A short calculation shows that the minimizer is given by

- If $s = y + z > 0$: optimal $x = \frac{s}{2} + 1$,
- If $s = y + z < 0$: optimal $x = \frac{s}{2} - 1$,
- And if $s = 0$: any $x \in [-1, 1]$ minimizes the function (we choose $x = 0$ for simplicity).

The same logic applies for updating y (with $s = x + z$) and z (with $s = x + y$). The code below implements this alternating minimization procedure.

python Copy

```python
import numpy as np

def phi(x):
    """Piecewise function φ(x)."""
    if x < -1:
        return (x + 1)**2
    elif x > 1:
        return (x - 1)**2
    else:
        return 0

def objective(x, y, z):
    """Objective function: -xy - yz - zx + φ(x) + φ(y) + φ(z)."""
    return -x*y - y*z - z*x + phi(x) + phi(y) + phi(z)

def update(s):
    """
    Given s (which will be y+z when updating x, etc.),
    returns the minimizer of φ(x) - s*x.
    """
    if s > 0:
        return s / 2 + 1
    elif s < 0:
        return s / 2 - 1
    else:
        return 0

def alternating_minimization(x0, y0, z0, tol=1e-6, max_iter=1000):
    """
    Alternates minimization over x, y, and z.
```

LLMs can reliably write code implementing non-trivial algorithms.

The answer is not always perfect (although it is perfect in this example), but you can use the generated code as a starting point and debug it.

LLMs often fail

ChatGPT o3-mini-high

I am having a Zoom meeting on March 5th with a colleague in Paris and Los Angeles. I am currently in Korea. When would be a good hour to have this meeting?

Reasoned for 2m 8s

A balanced option is to schedule the meeting for around 2:00 PM in Los Angeles on March 5. This works out as follows:

- **Los Angeles (PST, UTC−8):** 2:00 PM ← It's 2:00am. Calculation wrong.
- **Paris (CET, UTC+1):** 2:00 PM + 9 hours = 11:00 AM
- **Korea (KST, UTC+9):** 2:00 PM + 17 hours = 7:00 PM

This compromise lets your Los Angeles colleague meet in the early afternoon, your Paris colleague in mid-morning, and you in Korea in the early evening. Of course, if anyone prefers a slightly different time, you could adjust by an hour (for example, 3:00 PM in LA would shift Paris to noon and Korea to 8:00 PM), but 2:00 PM in Los Angeles tends to be a good starting point for everyone.

LLMs are excellent at understanding implied meaning or constraints through common sense reasoning. However, it still fails in some very basic steps.

Why? How can LLMs solve incredibly complex tasks while simultaneously failing at embarassingly simple ones?

Dual process theory

Dual process theory, popularized in the recent decades by Daniel Kahneman (Nobel economics laureate) and Amos Tversky, posits that our (human) thinking is governed by two distinct systems: a fast, automatic, and intuitive system (System 1) and a slower, more deliberate, and analytical system (System 2).

In this analogy, LLMs are good at System 1 thinking, but not System 2 thinking.

Why are LLMs good at System 1 thinking?

Why are LLMs not good at System 2 thinking despite the large-scale training?

LLMs are interpolative databases

François Chollet @fchollet

"It's autocomplete" is not a helpful analogy to understand LLMs. A LLM is more like a database that lets query information in natural language. You can query both knowledge, and "patterns" (associative programs seen in the training data, that can be applied to new inputs).

8:58 AM · Mar 18, 2023 · **384.5K** Views

François Chollet @fchollet · Mar 18, 2023

You can retrieve not just what was seen at training time, but arbitrary combinations of it.

It's an interpolative database and program store, with a natural language interface.

> **François Chollet** @fchollet · Aug 26, 2022
>
> Deep learning takes data points and turns them into a query-able structure that enables retrieval and interpolation between the points.
>
> You could think of it as a continuous generalization of database technology.

François Chollet @fchollet · Mar 18, 2023

When you scale this idea to "all the information on the Internet", you end up with something pretty powerful. Just like search, it doesn't have to be sophisticated to be impactful -- scale is the primary feature.

Hypothesis: System 1 thinking is about patter-matching past experience and knowledge, doing very basic (quick) mixing/interpolation of the set of relevant knowledge.

Pre-trained LLMs seem to behave in this way, as *interpolative databases*.

(LLMs clearly do more than merely memorizing raw facts, since they can solve new unseen tasks, provided that similar tasks were seen in training. So LLMs are not pure databases.)

https://x.com/fchollet/status/1637121320340299776

System 1 through a bag of heuristics

One view is that LLMs learn a large collection of heuristics, which have statistical correlations but do not represent the fundamental causal structure, and combine them.

Next-token-prediction training (analogous to imitation learning) does not force LLMs to learn the fundamental causal relation.

ARITHMETIC WITHOUT ALGORITHMS: LANGUAGE MODELS SOLVE MATH WITH A BAG OF HEURISTICS

Yaniv Nikankin[1]* Anja Reusch[1] Aaron Mueller[1,2] Yonatan Belinkov[1]
[1]Technion – Israel Institute of Technology [2]Northeastern University

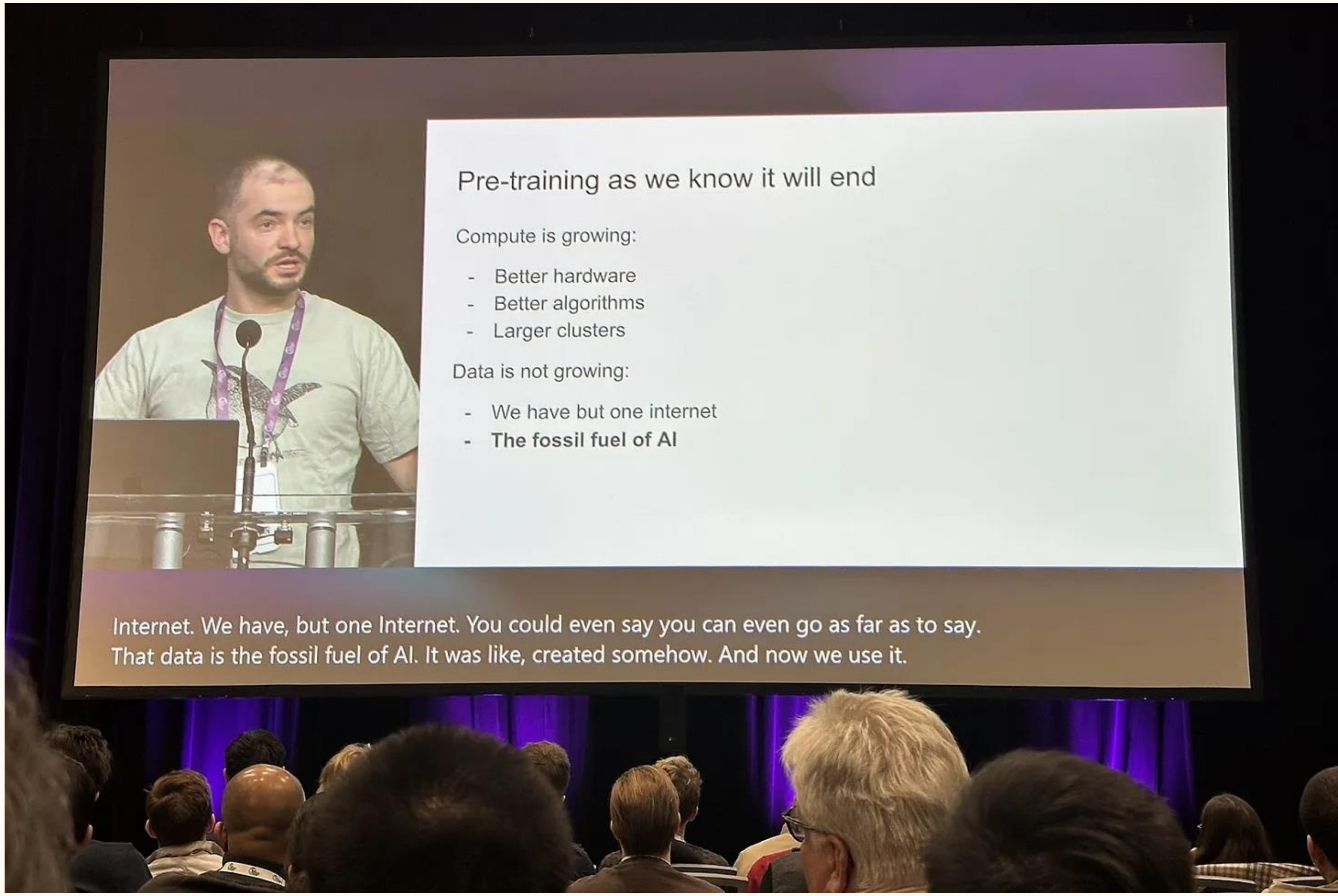
ABSTRACT

Do large language models (LLMs) solve reasoning tasks by learning robust generalizable algorithms, or do they memorize training data? To investigate this question, we use arithmetic reasoning as a representative task. Using causal analysis, we identify a subset of the model (a circuit) that explains most of the model's behavior for basic arithmetic logic and examine its functionality. By zooming in on the level of individual circuit neurons, we discover a sparse set of important neurons that implement simple heuristics. Each heuristic identifies a numerical input pattern and outputs corresponding answers. We hypothesize that the combination of these heuristic neurons is the mechanism used to produce correct arithmetic answers. To test this, we categorize each neuron into several heuristic types—such as neurons that activate when an operand falls within a certain range—and find that the unordered combination of these heuristic types is the mechanism that explains most of the model's accuracy on arithmetic prompts. Finally, we demonstrate that this mechanism appears as the main source of arithmetic accuracy early in training. Overall, our experimental results across several LLMs show that LLMs perform arithmetic using neither robust algorithms nor memorization; rather, they rely on a "*bag of heuristics*". [1]

Y. Nikankin, A. Reusch, A. Mueller, and Y. Belinkov, Arithmetic without algorithms: Language models solve math with a bag of heuristics, *ICLR*, 2025.

The era of scaling pre-training is over

Scaling pre-training (more data, large neural network, more compute) made everything better, until it stopped.



I. Sutskever, Test of Time Award Talk, *NeurIPS*, Dec. 13, 2024.

The era of scaling pre-training is over

Scaling from GPT-3 (and GPT-3.5) to GPT-4 lead to remarkable improvements.

Scaling from GPT-4 to GPT-4.5 did not. LLM pre-training scaling has plateaued.

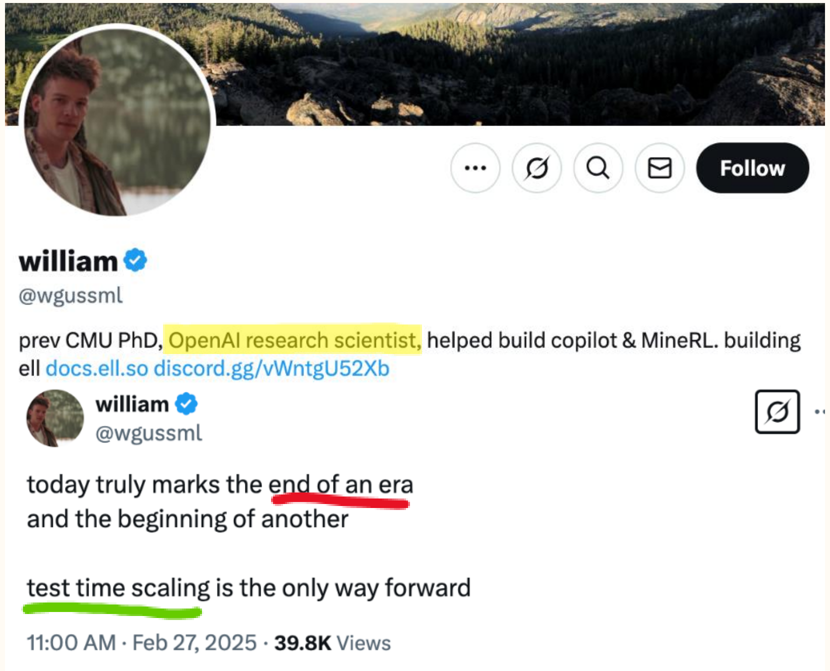
4 Preparedness Framework Evaluations

GPT-4.5 is not a frontier model, but it is OpenAI's largest LLM, improving on GPT-4's computational efficiency by more than 10x. While GPT-4.5 demonstrates increased world knowledge, improved writing ability, and refined personality over previous models, it does not introduce

OpenAI GPT-4.5 System Card, Feb. 27, 2025. (Retrieved on Mar. 01, 2025.)
(The quote seems to have been removed from the system card since the retrieval.)

The era of scaling pre-training is over

Following the release of the GPT-4.5 System Card, many OpenAI researchers (presumably with inside knowledge) confirmed that scaling pre-training is saturating. We need a different strategy to make the next step.



And that strategy is **RL**, **search**, and **test-time scaling**.

https://x.com/wgussml/status/1895187231666774377

RL is starting to work

Before 2010, many ML researchers would say “RL doesn’t work” or “RL is a scam”.

- “Autonomous helicopter flight via reinforcement learning” (*NeurIPS* 2003) was somewhat impressive, but not that amazing. Would certainly not be deployed.

But now, RL is starting to work.

- DeepMind’s DQN paper solving Atari Games (*Nature* 2015) was a bit wakeup call.
- AlphaGo for Go (*Nature* 2016), AlphaStar for StarCraft II (*Nature* 2019), Pluribus for Texas hold’em poker (*Science* 2019), Cicero for the Diplomacy boardgame (Science 2022) are all success stories of RL achieving super-human performance in games.
- Autonomous taxis service several cities, and customer satisfaction is very high.
    - Disclaimer: It is unclear how much RL is being used, even though the companies claim to do so.
- Reinforcement learning with human feedback (RLHF) is an indispensable part of LLM alignment
    - Disclaimer: Some argue that RLHF is not “true RL”.
- Many robotics companies showcase impressive demos of humanoid or drone robots.
    - Disclaimer: It is unclear how much of the demos are hand-crafted or outright human controlled.

RL **is** creative

There is a misconception that AI simply regurgitates training data, but this is false. AI with RL has shown creativity.

“Reinforcement learning is actually creative. Every single stunning example of creativity in AI comes from a reinforcement learning system. For example, AlphaZero has invented a whole new way of playing a game that humans have perfected for thousands of years. It is reinforcement learning that can come up with creative solutions to problems — solutions which we might not be able to understand at all.”

— Ilya Sutskever —

February 27, 2023

I. Sutskever, Opening remarks: Confronting the possibility of AGI, *San Francisco Alignment Workshop*, Feb. 27 2023. https://youtu.be/OPZxs6IXH00?si=iSj7WTSk2Yid5pek

RL’s creativity changed chess

“The neural nets have improved our understanding of the game immensely. … after AlphaZero came out in late 2018. … And it just made us understand the game a lot better. [Describing a strategy on using pawns.] This is something that humans didn't really do”

— Magnus Carlsen —

M. Carlsen, *Joe Rogan Experience Episode 2275*, Feb. 20, 2025.

RL’s creativity changed Poker

“The game of poker the way it looked when I started playing in the late 90s is very different to what you see today … a lot of the top players at the very highest level use [AI] to improve their game”

— Daniel Negreanu—

“There were a few things that the humans walked away from, but [describing the overbet strategy] was the number one thing that the humans walked away from the competition saying like, we need to start doing this.”

— Noam Brown —

D. Negreanu, *CBS Sports Interview*, Jul. 2, 2019.
N. Brown, *Lex Fridman Podcast Episode 344*, Dec. 6, 2022.

RL’s creativity changed Go

“Game records from before AlphaGo are completely different from those of today. The old records now have historical value and not for studying Go. … One disappointing aspect is that AI Go feels like you’re just looking at an answer key.”

— Sedol Lee —

S. Lee, The Dong-A Ilbo Interview, Mar. 19, 2024.

RL using System 2 thinking

The success stories of RL in games combines self-play (RL) and a search mechanism (System 2 thinking).

- The raw neural network proposes reasonable actions with one immediate evaluation of the neural network. This is like humans' instinctive System 1 thinking.
- Then, with mechanisms like neural-guided Monte Carlo tree search, deliberate on which of the possible actions are good. This is like humans' System 2 thinking.

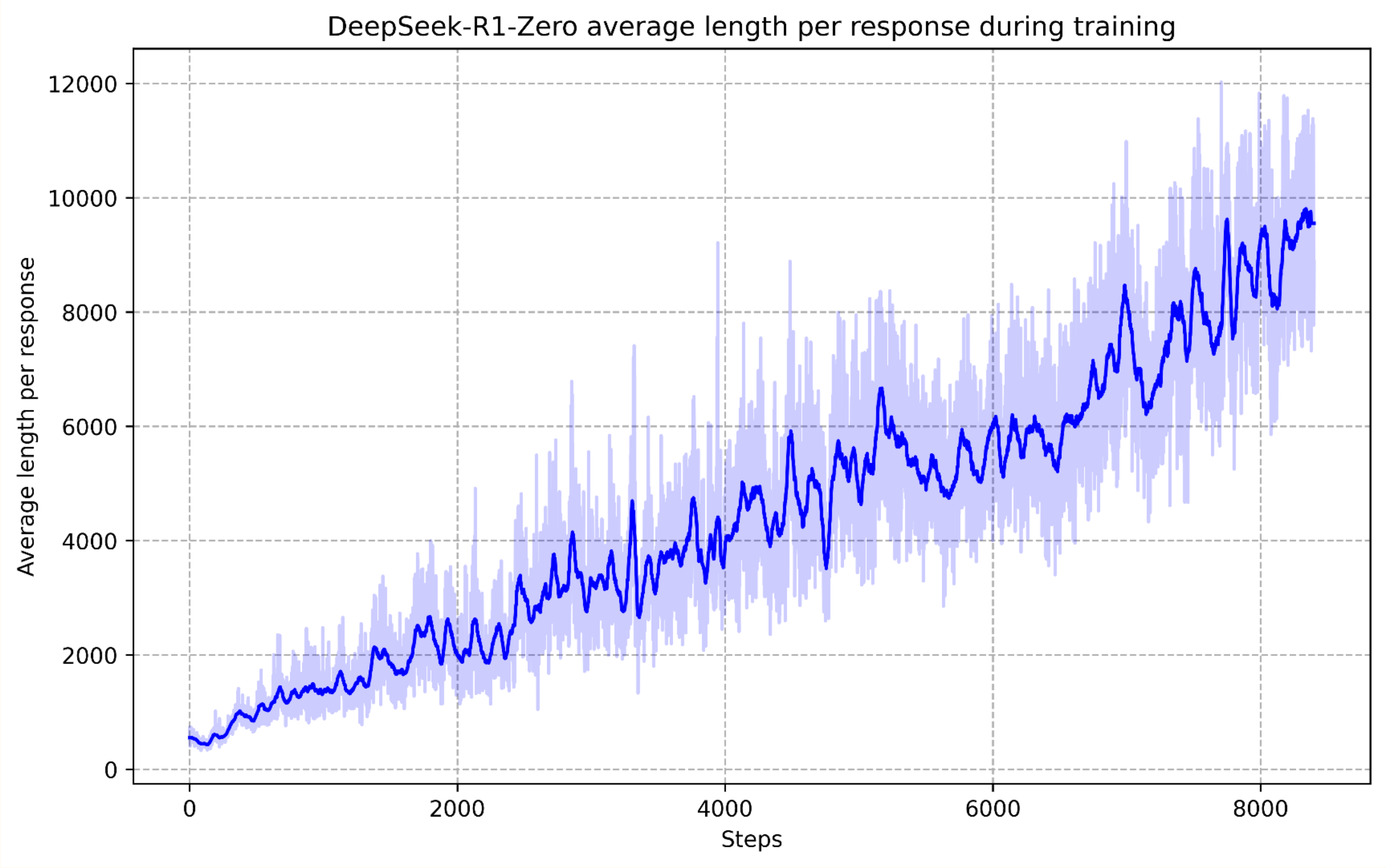
Since the System 2 thinking happens when the model is being deployed, scaling this deliberation capability is called test-time scaling.

The DeepSeek-R1 pipeline

Train strong baseline model. (The subsequent RL only works if the baseline LLM is already very strong.)

RL-train on producing a correct answer on verifiable domains such as coding or math.

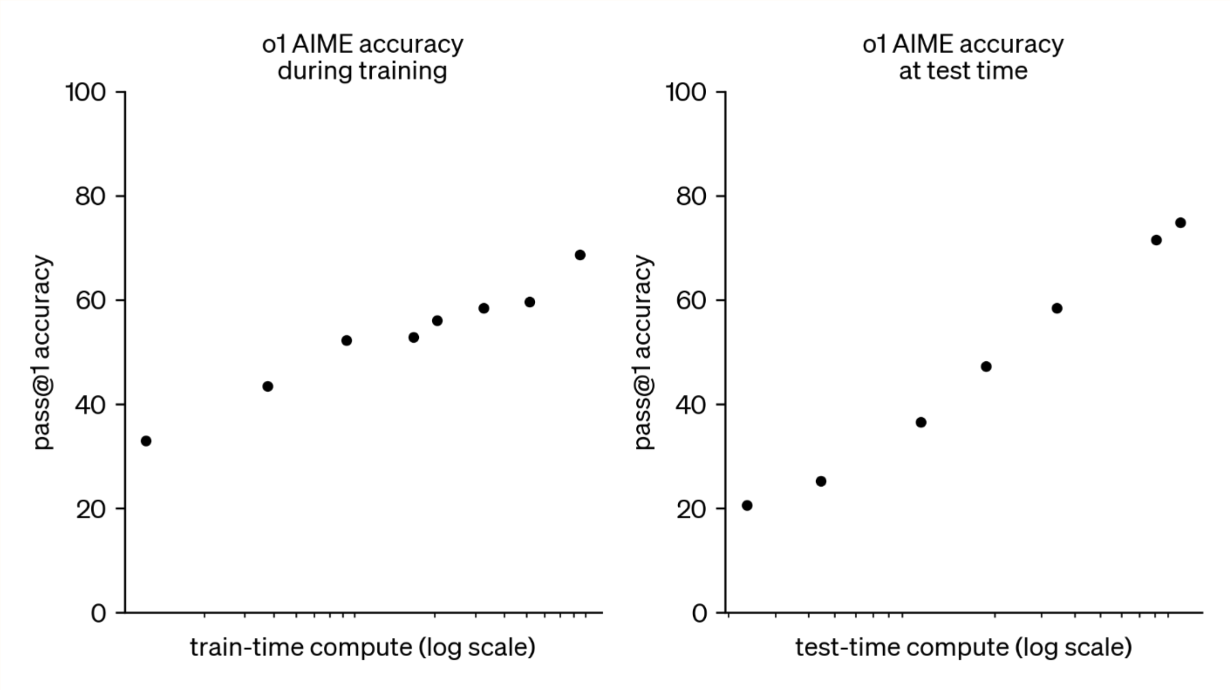
Emergent ability: LLM learns to utilize CoT.

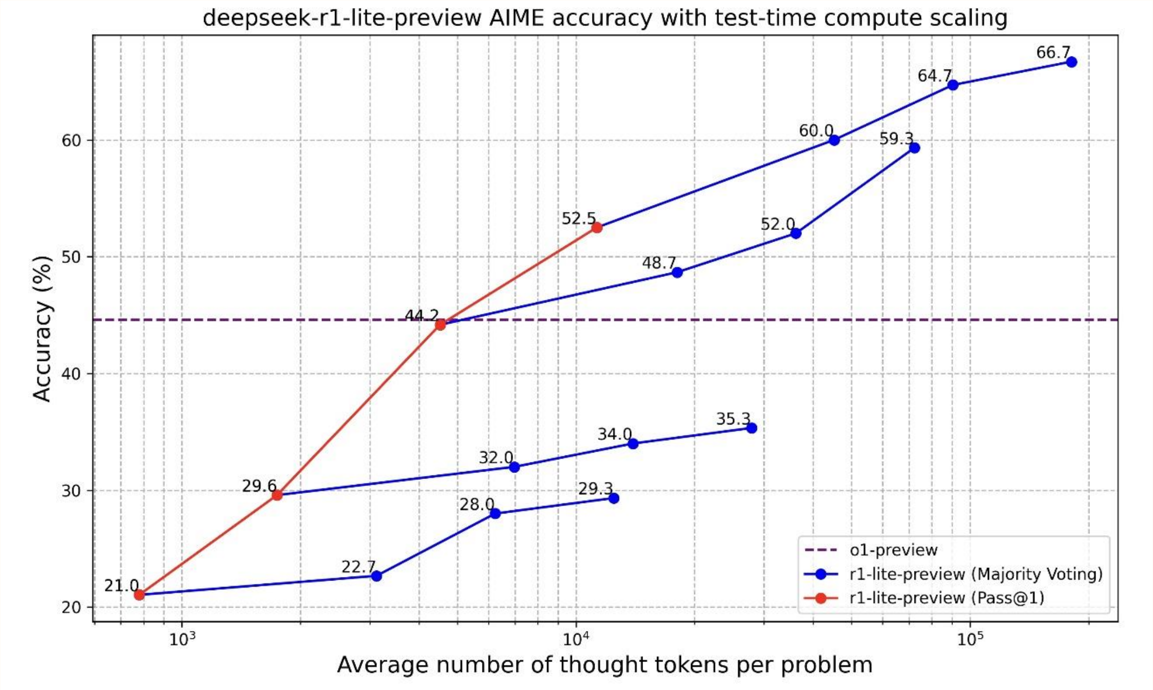


DeepSeek-AI, DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, *arXiv*, Jan. 2025.

Through RL, LLMs learn to deliberate

Using longer CoT (thinking time) OpenAI o1 and DeepSeek-R1 significantly improve at mathematics problem solving (AIME).





Roughly, x-axis is thinking time. We will return to these plots later.

OpenAI, https://openai.com/index/learning-to-reason-with-llms, Sept. 12, 2024.
DeepSeek, https://x.com/deepseek_ai/status/1859200149844803724, Nov. 20, 2024.

Why is RL promising?

Reason 1: Through RL, LLMs can learn how to use CoT, which roughly means talking to itself before answering the question. The human's internal thought process leading to the text is mostly unavailable in pre-training data, but we can replicate it with RL.

- Deepseek R1 and OpenAI o1 (and o3) has clearly demonstrated this. Refined executions of this will certainly bring further improvements.

Reason 2: RL may be the key to finding the correct causal relation and shed the misunderstandings among the heuristics.

- Less explored function of RL. May lead to LLMs discovering "true" generalizable knowledge.

Human learn correct causal relations through actions and rewards

We receive explicit supervision or observations we can imitate. We then act in the world and receive a reward. The active practice reinforces our brains' memory, but crucially, it also allows you to discover and correct misunderstandings.

Example) Imagine you see a basketball move. It looks like:

1. Dribble fast and pass the defender.
2. Lay up.

You imitate it, but you realize the defender keeps up with you and it doesn't work.

Through practice, you realize the correct steps are:

1. Fake out the defender with an eye fake.
2. Dribble fast and pass the defender.
3. Lay up.

Without the fake, the play doesn't work.
To learn this, observation is not enough.
You must try out the play yourself.

SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training

Tianzhe Chu ♠ * Yuexiang Zhai ♥ ♣ * Jihan Yang ♦ Shengbang Tong ♦
Saining Xie ♣ ♦ Dale Schuurmans ♣ Quoc V. Le ♣ Sergey Levine ♥ Yi Ma ♠ ♥

Hypothesis: Current LLMs do pattern matching, and reasoning mistakes arise when the pattern matching is mis-applied. Maybe, RL will allow LLMs to transcend pattern maching and learn the correct generalizable knowledge.

T. Chu, Y. Zhai, J. Yang, S. Tong, S. Xie, D. Schuurmans, Q. V. Le, S. Levine, and Y. Ma, SFT memorizes, RL generalizes: A comparative study of foundation model post-training, *ICML*, 2025.

Abstract

Supervised fine-tuning (SFT) and reinforcement learning (RL) are widely used post-training techniques for foundation models. However, their respective role in enhancing model generalization remains unclear. This paper studies the comparative effect of SFT and RL on generalization and memorization, focusing on text-based and visual environments. We introduce `GeneralPoints`, an arithmetic reasoning card game, and also consider `V-IRL`, a real-world navigation environment, to assess how models trained with SFT and RL generalize to unseen variants in both textual and visual domains. We show that RL, especially when trained with an outcome-based reward, generalizes in both the rule-based textual and visual environments. SFT, in contrast, tends to memorize the training data and struggles to generalize out-of-distribution in either scenario. Further analysis reveals that RL improves the model's underlying visual recognition capabilities, contributing to its enhanced generalization in visual domains. Despite RL's superior generalization, we show that SFT is still helpful for effective RL training: SFT stabilizes the model's output format, enabling subsequent RL to achieve its performance gains. These findings demonstrate the advantage of RL for acquiring generalizable knowledge in complex, multi-modal tasks.

Summer of RL and AI

We have before us the summer of RL and AI.

Through RL, search, and test-time scaling, AI will make at least one big step of progress.

Whether this progress will take us to AGI or will stop after one step is to be seen.