

Lecture 1: Overview

RLHFBOOK.COM

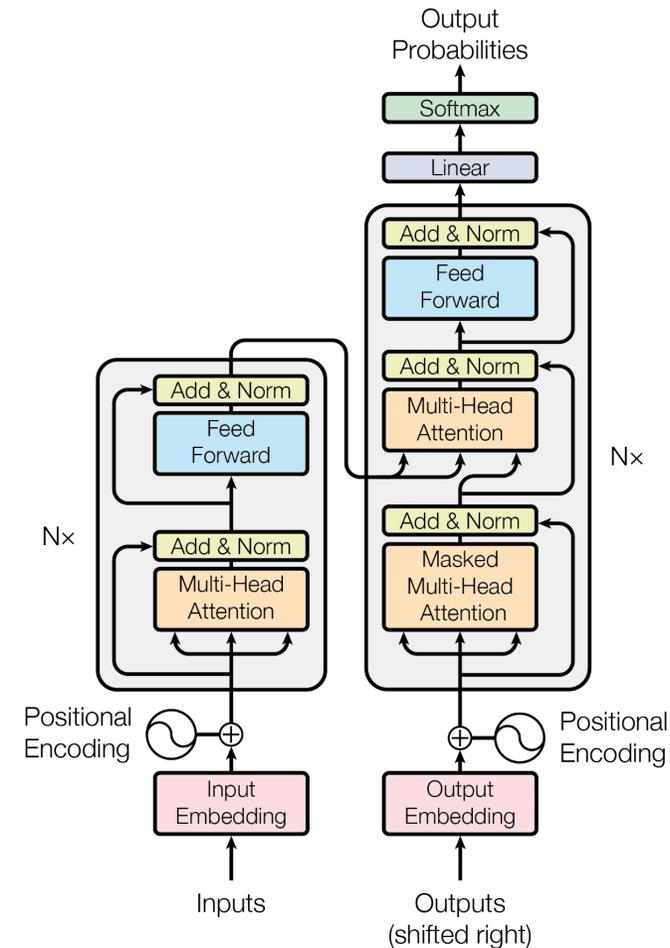
Nathan Lambert

Course on RLHF and post-training. Chapters 1-3

What is a language model?

Core properties:

- A language model assigns probabilities to text.
- Chunks of words are broken down as **tokens**, which are the internal representation of the model.
- Given previous tokens, it predicts the next token. Repeating this produces a completion one step at a time (this is called **autoregressive**).

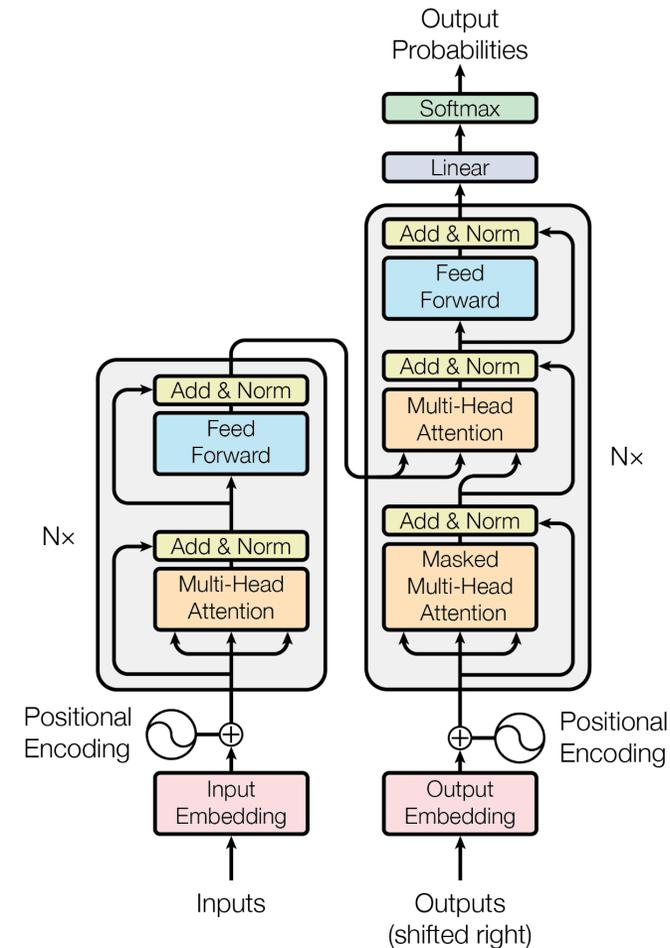


The original model architecture diagram for the Transformer. 2017.

What is a (modern) language model?

Modern language models:

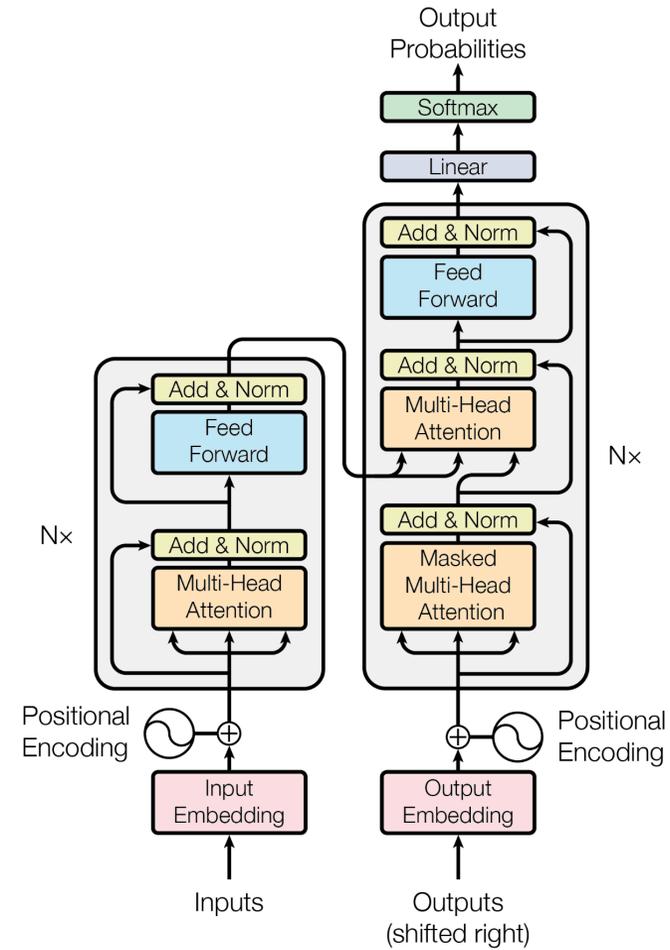
- Have billions to trillions of parameters.
- Largely downstream of the Transformer architecture, which popularized the use of the **self-attention** mechanism along with fully-dense layers.
- Predict and work over much more than text: Gemini and ChatGPT work with images, audio, and video.



The original model architecture diagram for the Transformer. 2017.

2017: The Transformer is born

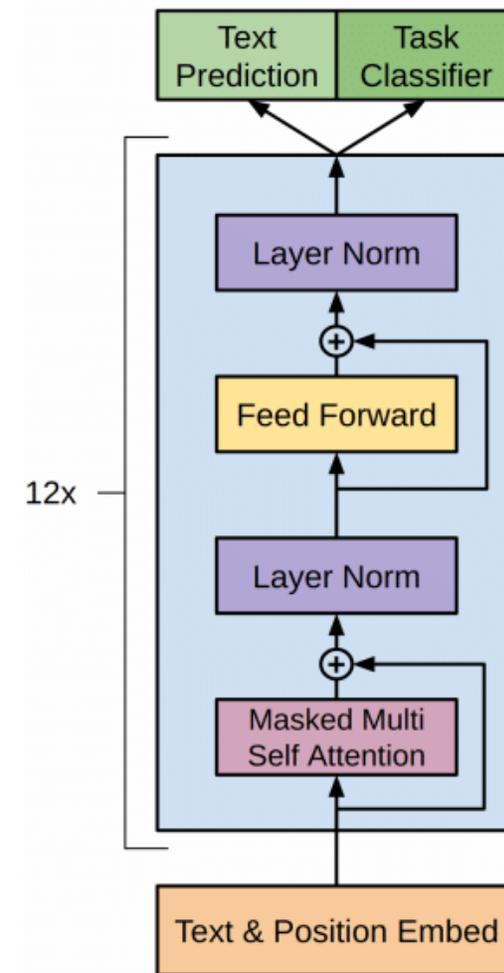
- 2017: the Transformer is born



The original model architecture diagram for the Transformer. 2017.

2018: GPT-1, ELMo, and BERT

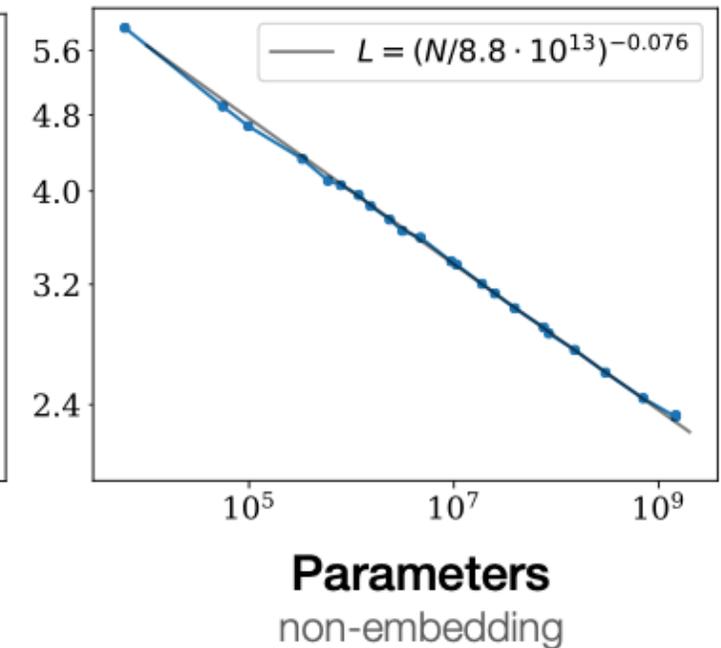
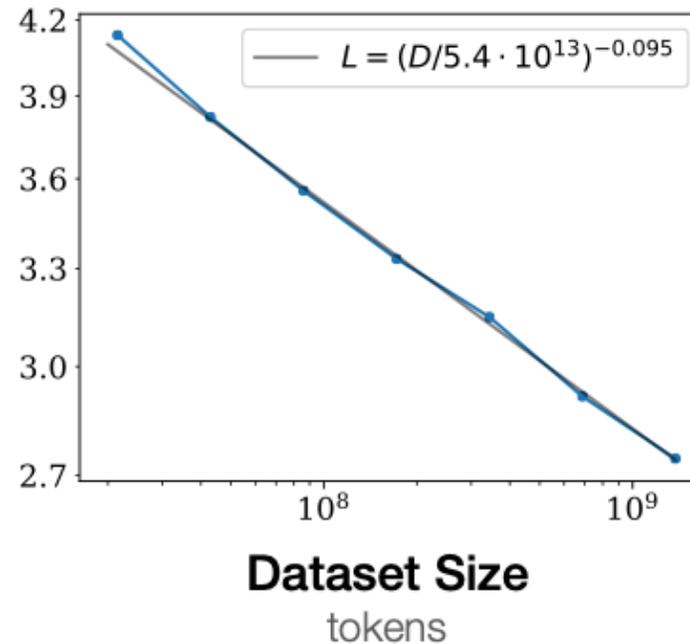
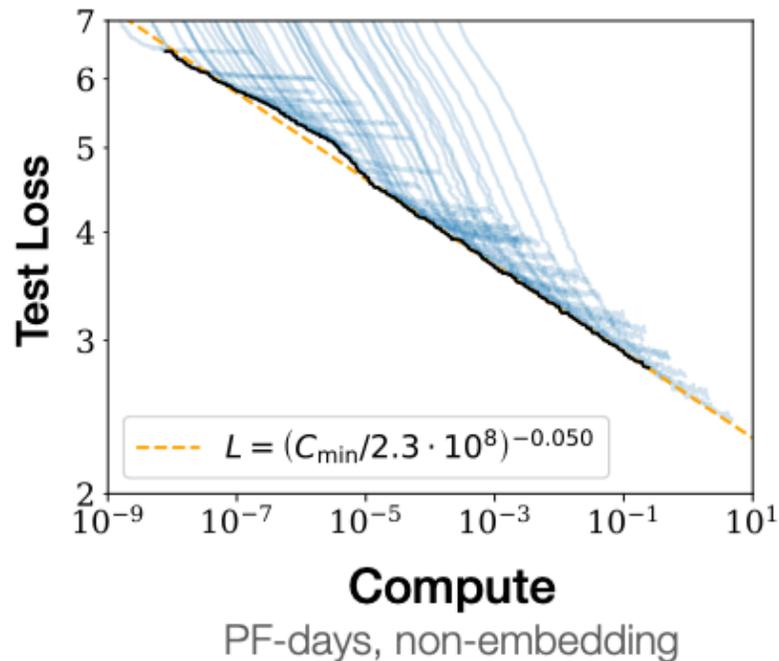
- 2017: the Transformer is born
- **2018:** GPT-1, ELMo, and BERT released



The language model architecture for GPT-1. 2018.

2019: GPT-2 and scaling laws

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- **2019: GPT-2 and scaling laws**



Kaplan et al., 2020; Radford et al., 2019

2020: GPT-3 surprising capabilities

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- **2020: GPT-3 surprising capabilities**

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ← examples
4 plush girafe => girafe peluche ← examples
5 cheese => ..... ← prompt
```

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



GPT-3 was known for expanding the idea of in-context learning and few-shot prompting. Screenshot from the paper.

2021: Stochastic Parrots

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- **2021: Stochastic Parrots**

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

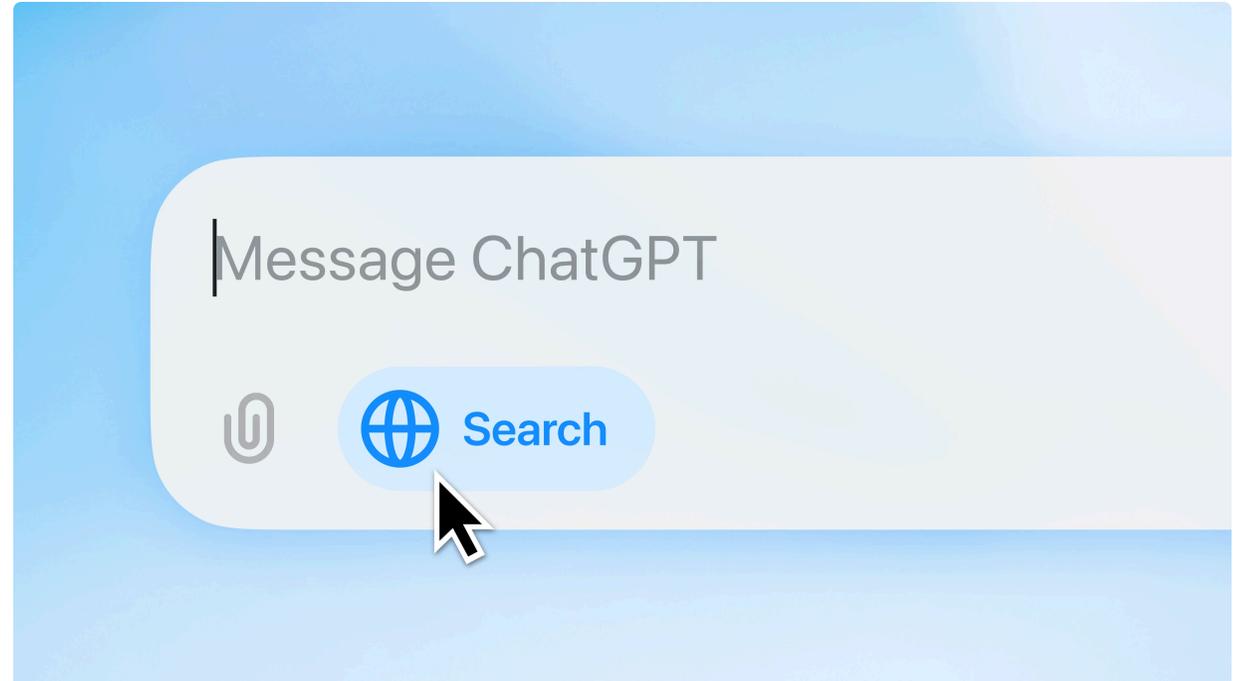
The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized com-

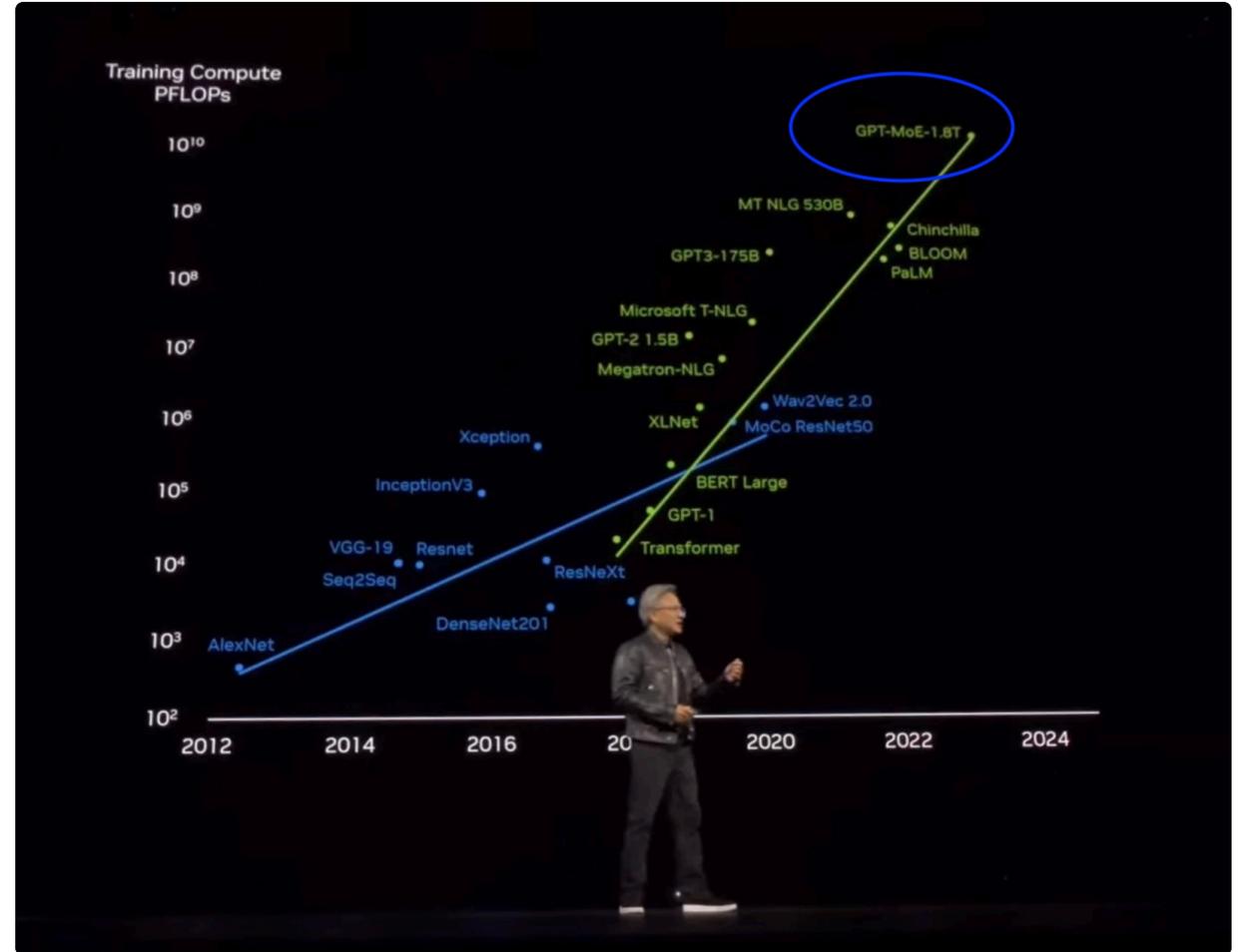
2022: ChatGPT

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- 2021: Stochastic Parrots
- **2022: ChatGPT**



2023: GPT-4 and frontier-scale

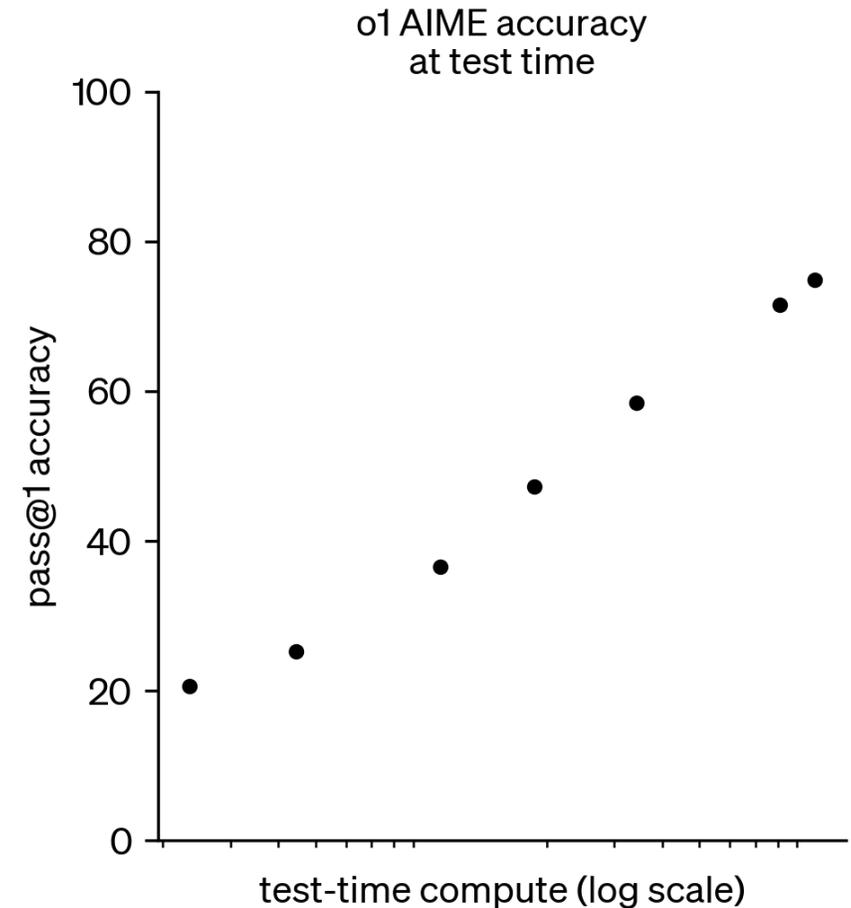
- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- 2021: Stochastic Parrots
- 2022: ChatGPT
- **2023: GPT-4 and frontier-scale**



An image where Nvidia CEO Jensen Huang supposedly leaked that GPT-4 was an ~2T parameter MoE model.

2024: o1 and reasoning models

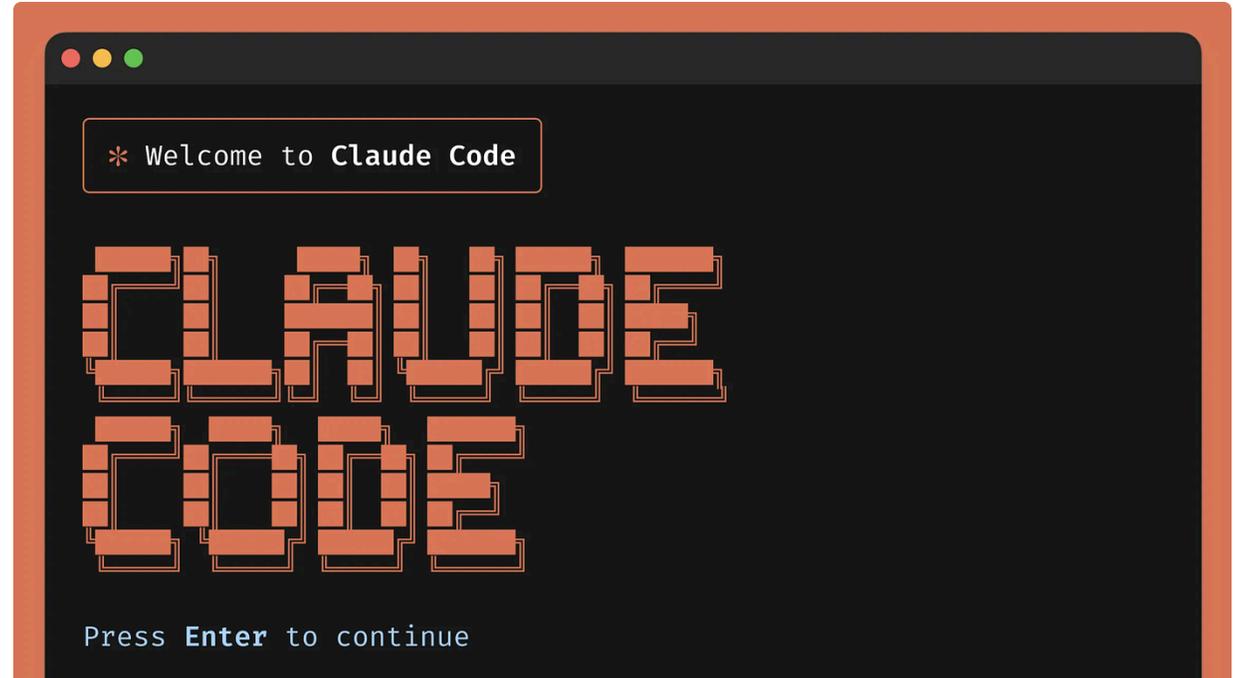
- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- 2021: Stochastic Parrots
- 2022: ChatGPT
- 2023: GPT-4 and frontier-scale
- **2024: o1 and reasoning models**



The famous test-time scaling plot from OpenAI's o1 announcement.

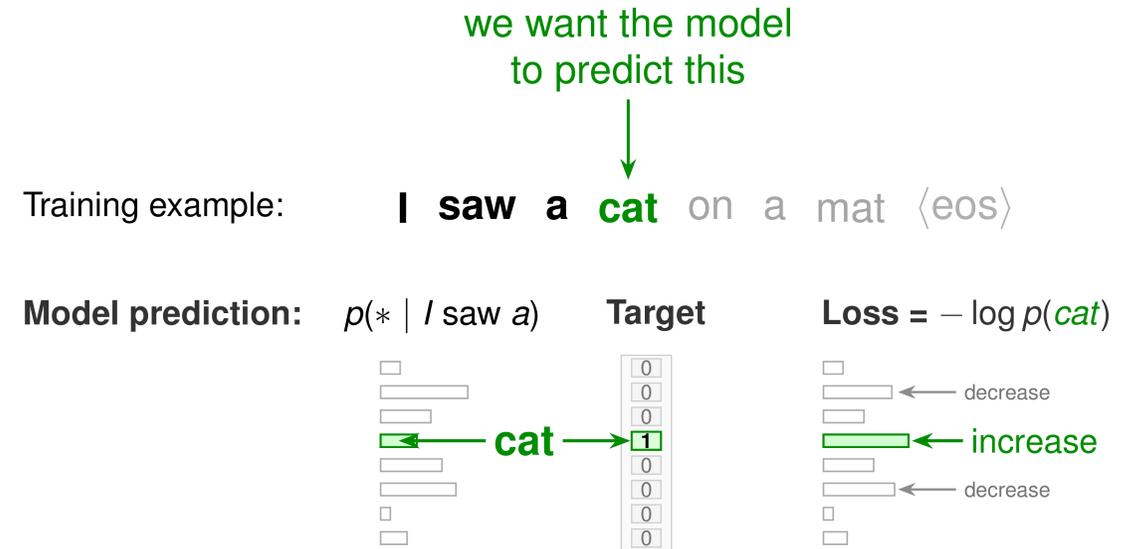
2025: o3, Claude Code, and agents

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- 2021: Stochastic Parrots
- 2022: ChatGPT
- 2023: GPT-4 and frontier-scale
- 2024: o1 and reasoning models
- **2025: o3, Claude Code, and agents**



Pretraining: next-token prediction

- Train on trillions of tokens of text from the web, books, code, and documents
 - Models are often trained on 5-50+ trillion tokens
 - 1T of text tokens is about 3-5 TB of data
 - Labs gather and filter 10-20X more data than is used for the model
 - Total data funnel targeted for models is on the order of petabytes
- Objective: predict the next token in each sequence
- Result: Incredible, flexible, useful models



A base model completes text

After pretraining we are left with a glorified autocomplete model, for example:¹

USER

The president of the United States in 2006 was

ASSISTANT (Llama 3.1 405B Base)

George W. Bush, the governor of Florida in 2006 was Jeb Bush, and John McCain was an Arizona senator in 2006...

¹: Base models are also becoming more flexible through midtraining and better data mixtures.

Post-training makes it answer like a chatbot

The earliest forms of modern post-trained (or RLHF-tuned) models shifted the continuation format to always conforming to the “answering a question style.” An example of what early conversational models looked like is below:

USER

The president of the United States in 2006 was

ASSISTANT (Tülu 3 405B)

George W. Bush was the president of the United States in 2006. He served two terms in office, from January 20, 2001, to January 20, 2009.

Intuition for training stages

How the different training stages change the model:

- **Pretraining**: builds the model's world knowledge, language fluency, and broad capabilities
- **Instruction tuning / SFT**: teaches the model to answer in a question-answer format and often teaches it to repeat specific token sequences
- **Preference tuning / RLHF**: uses a contrastive loss to modify completions as a whole, making the model richer and more flexible
- **RLVR**: enhances the model's ability on verifiable questions, which can translate into more complex, agentic behaviors

Intuition for training stages

How the different training stages change the model:

- **Pretraining**: builds the model's world knowledge, language fluency, and broad capabilities
- **Instruction tuning / SFT**: teaches the model to answer in a question-answer format and often teaches it to repeat specific token sequences
- **Preference tuning / RLHF**: uses a contrastive loss to modify completions as a whole, making the model richer and more flexible
- **RLVR**: enhances the model's ability on verifiable questions, which can translate into more complex, agentic behaviors

What this means for post-training:

- RL-based post-training gives the model richer, response-level, contrastive feedback that makes it more flexible and engaging
- SFT is still needed as a foundation for maximum performance

ChatGPT was when RLHF made the models even easier to use

Model responses evolved quickly to have:

- Better **format**: direct, conversational answers
- Better **style**: helpful, concise, markdown, etc.
- Better **product**: people could actually use it every day

USER

I'm giving a talk on RLHF tomorrow. Can you help me structure it?

ASSISTANT

Absolutely — here's a simple structure:

1. Start with the basics

- What a language model is
- Why next-token prediction is not enough

2. Explain post-training

- Instruction tuning
- Preference tuning / RLHF
- RLVR for verifiable tasks

3. End with why it matters

- Better products like ChatGPT
- Better reasoning in models like o1

...

So what is reinforcement learning from human feedback (RLHF) anyways?

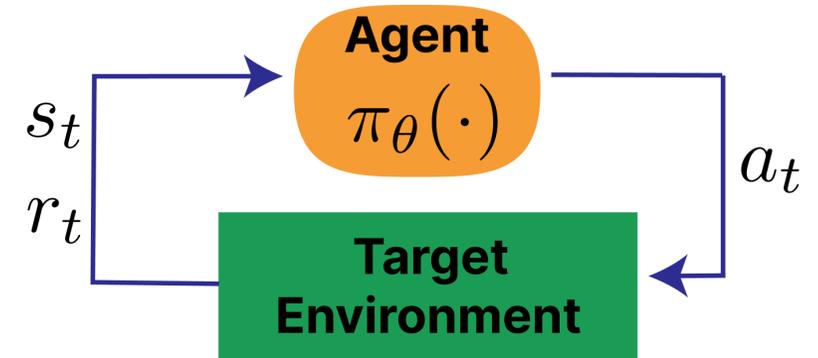
Classical Reinforcement Learning (RL)

A reinforcement learning problem is often written as a **Markov Decision Process (MDP)**:

- state space \mathcal{S} , action space \mathcal{A}
- transition dynamics $P(s_{t+1} \mid s_t, a_t)$
- reward function $r(s_t, a_t)$ and discount γ
- optimize cumulative return over a trajectory

MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$

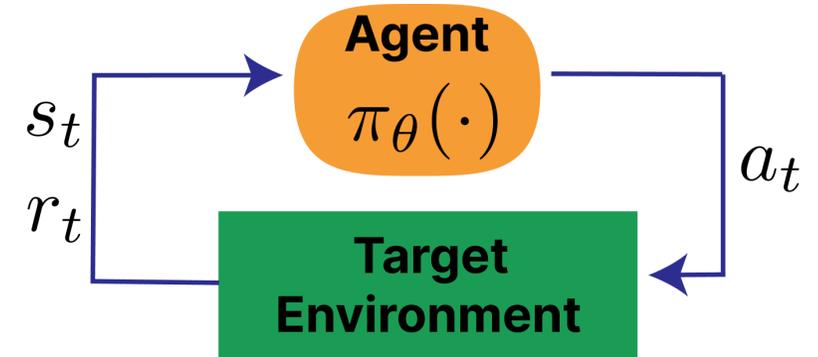
$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \right]$$



RL in plain language

Reinforcement learning basics:

- Reinforcement learning is **trial-and-error learning**
Balancing exploration and exploitation across long-term rewards
- **State**: the current situation the agent is in
- **Action**: what the agent does next
- **Reward**: the signal for how good that action was
- **Policy**: the strategy for choosing actions
- **Trajectory**: a sequence of states, actions, and rewards: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$
- **Trajectory probability**:
$$P(\tau \mid \pi) = p(s_0) \prod_{t=0}^T \pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$



A simple RL example: thermostat

The agent learns over many episodes when to turn the heater on or off

- **State:** the current room temperature
- **Action:** turn the heater on or off
- **Reward:** positive when the room stays near the target temperature
- **Policy:** the rule for deciding what to do next

Example policy:

$$\pi(a_t = \text{on} \mid s_t) = \begin{cases} 1 & \text{if } s_t < 70^\circ\text{F} \\ 0 & \text{otherwise} \end{cases}$$

CartPole: a standard RL task

- **State:** cart position, velocity, pole angle, angular velocity

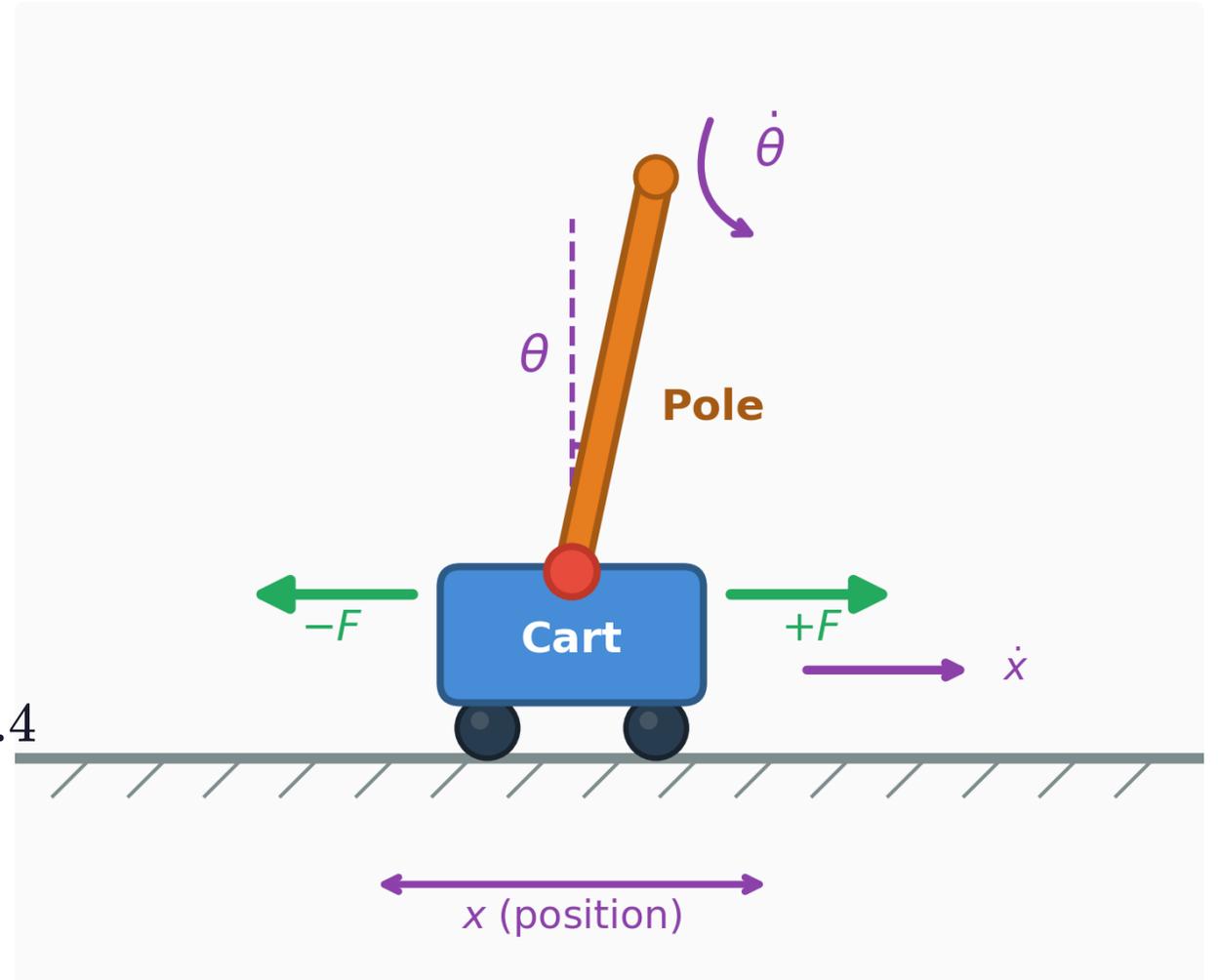
$$s_t = (x_t, \dot{x}_t, \theta_t, \dot{\theta}_t)$$

- **Action:** push the cart left or right

$$a_t \in \{\text{left}, \text{right}\}$$

- **Reward:** +1 for every step the pole stays upright

$$r(s_t, a_t) = \begin{cases} 1 & \text{if } |\theta_t| < 12^\circ \text{ and } |x_t| < 2.4 \\ 0 & \text{otherwise (episode ends)} \end{cases}$$



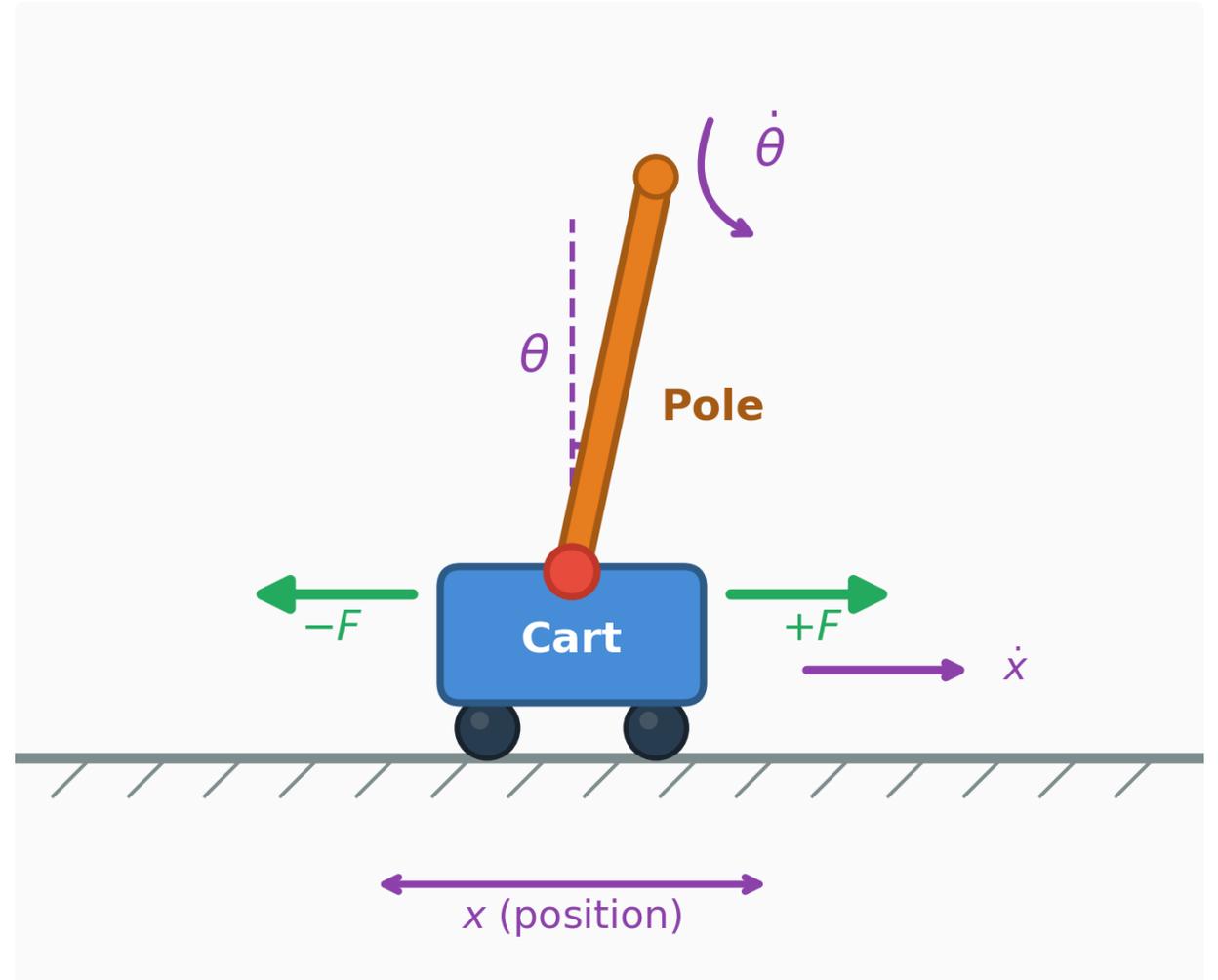
CartPole: the dynamics

Each action changes the physics of the system. The full state update:

$$\ddot{x}_t = \frac{F + m_p l (\dot{\theta}_t^2 \sin \theta_t - \ddot{\theta}_t \cos \theta_t)}{m_c + m_p}$$
$$\ddot{\theta}_t = \frac{g \sin \theta_t - \cos \theta_t \cdot \ddot{x}_t}{l}$$

Where m_c is the cart mass, m_p is the pole mass, l is the pole length, g is gravity, and F is the applied force.

This is why classical RL is a **multi-step control problem** — each action changes the next state, and rewards accumulate across a trajectory.



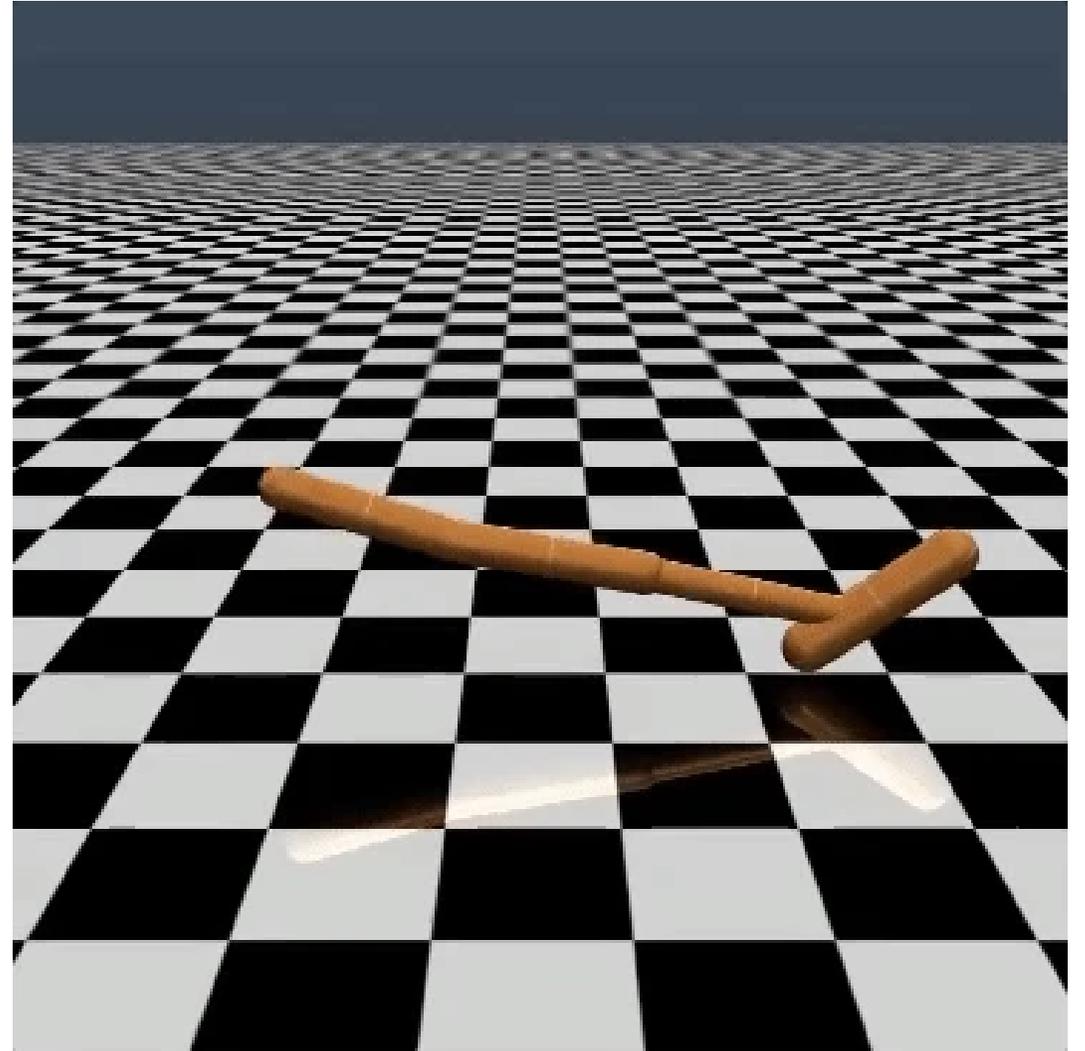
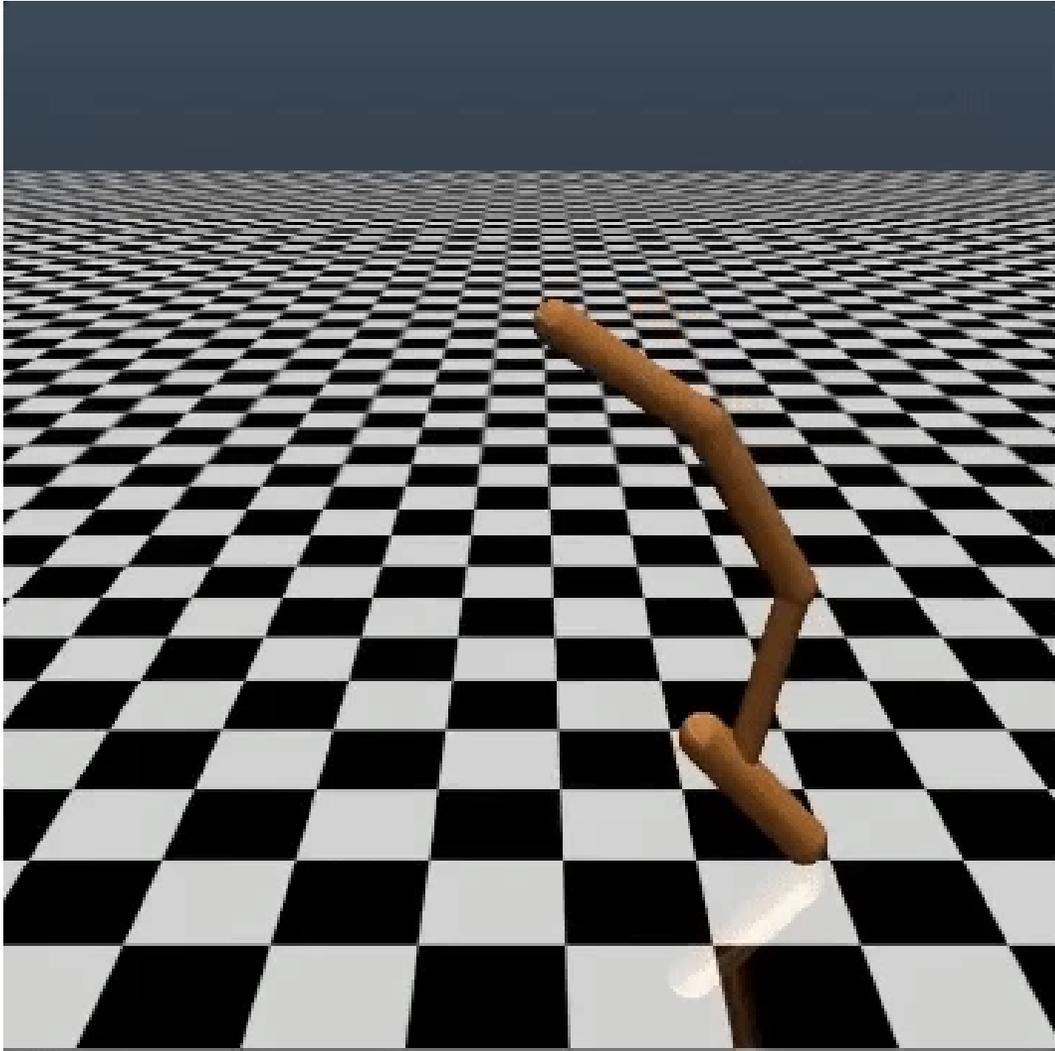
Why did people make RLHF?

In classical RL, the reward function is known — CartPole gives +1 per step. But for many tasks the reward is **hard to write down**:

- It's easy to judge which poem is better, but hard to write a masterpiece
- It's easy to spot a helpful answer, but hard to specify what “helpful” means as a formula
- Pretraining optimizes **next-token prediction** — the most likely continuation isn't always the most useful one

RLHF lets us optimize for behavior we can **evaluate** even when we cannot easily **specify** the reward.

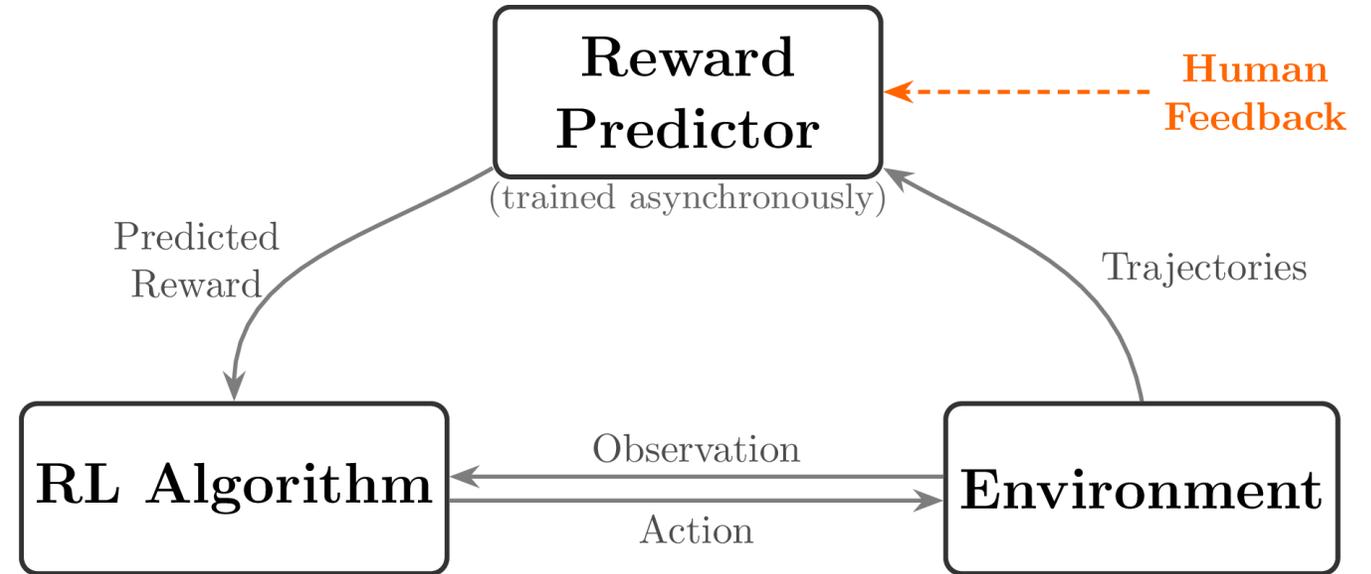
Which is the better backflip?



Christiano et al., 2017

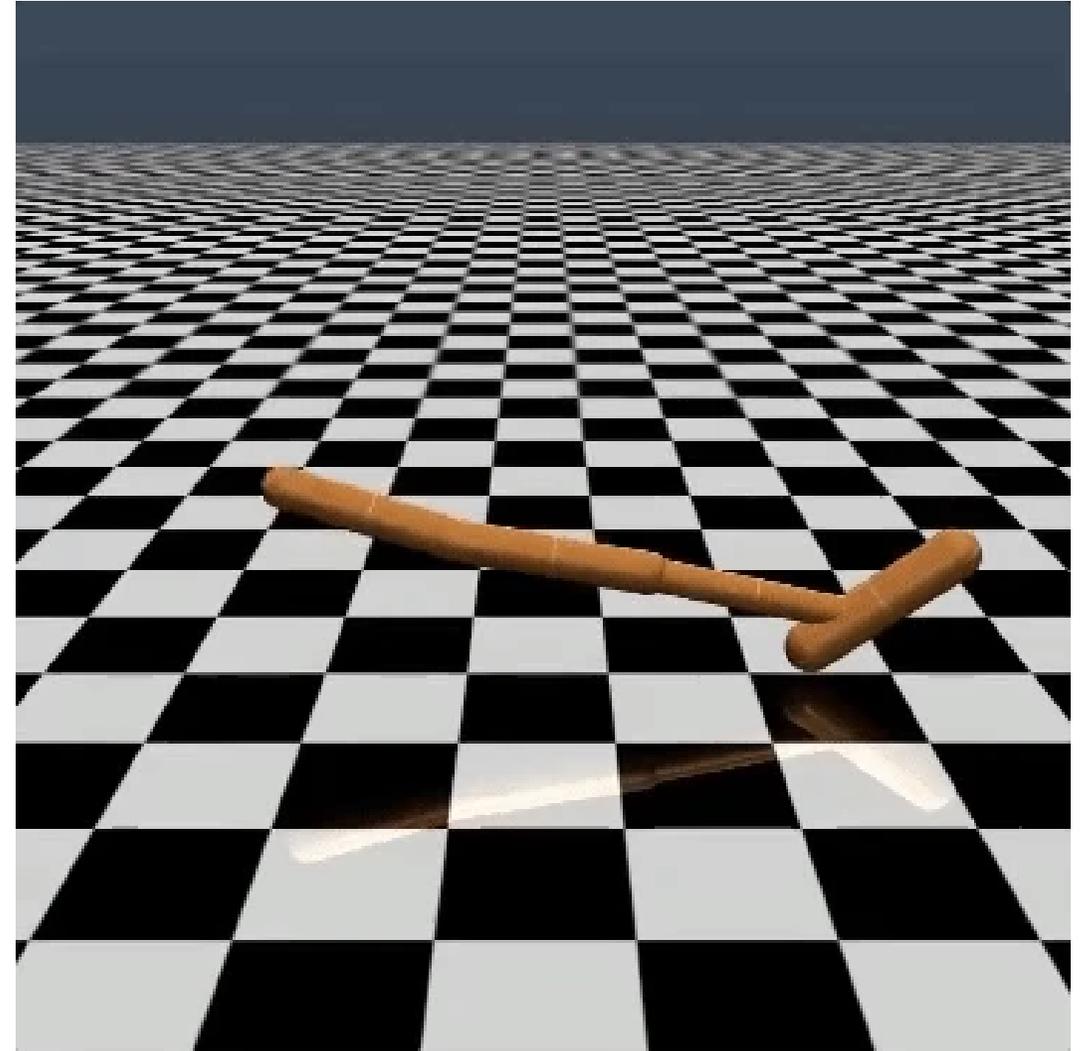
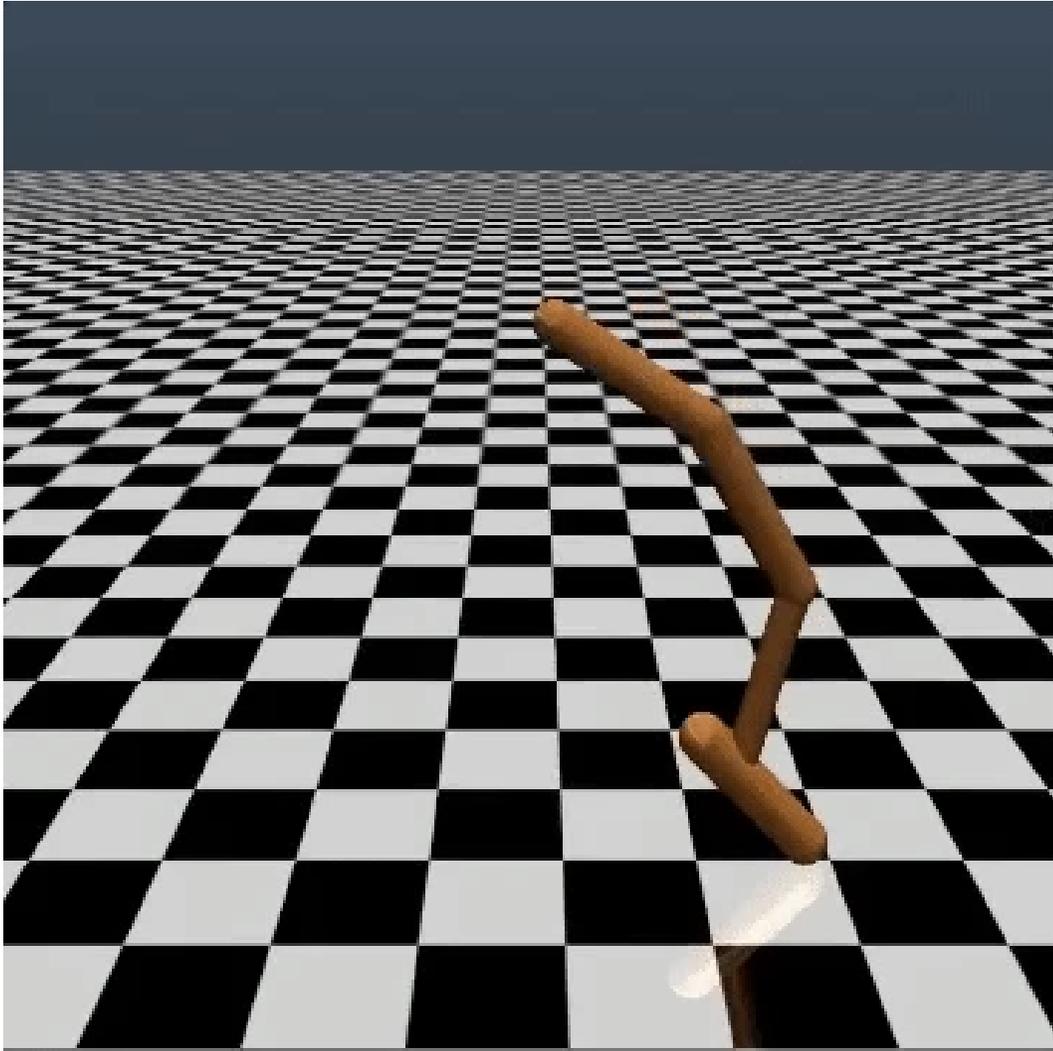
RLHF before language models

- **TAMER** (Knox & Stone, 2008) — humans score agent actions to learn a reward
- **Christiano et al. 2017** — RLHF on Atari trajectory preferences
- **Ziegler et al. 2019** — first RLHF on language models

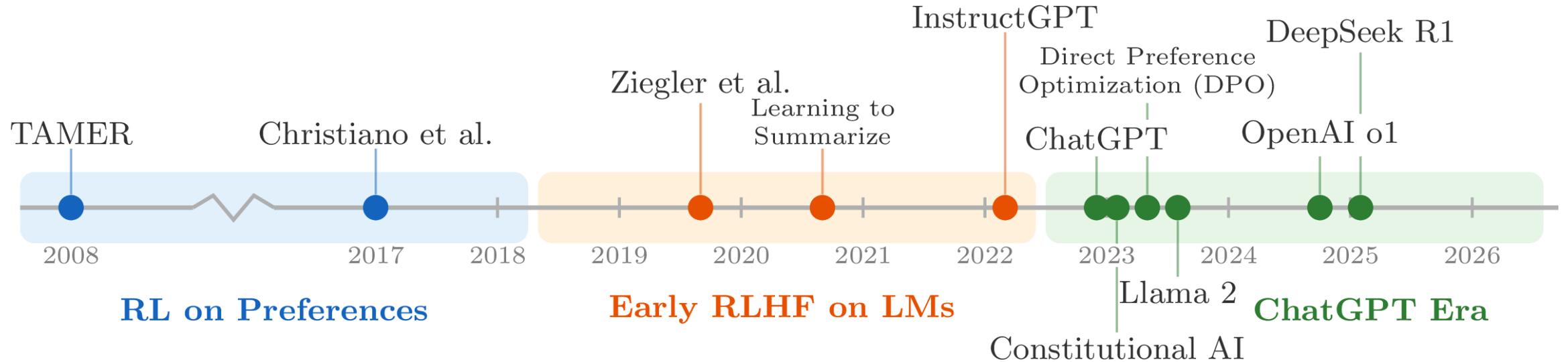


A recreation of the system diagram from Christiano et al. 2017.

Left: Human feedback; Right: Hand-designed reward function



The path to modern RLHF



- **Ziegler 2019** (Ziegler et al., 2019) — first RLHF on language models
- **InstructGPT** (Ouyang et al., 2022) — the canonical RLHF recipe behind ChatGPT
- **Constitutional AI** (Bai et al., 2022) — Introduced early methods for AI feedback in Claude
- **DPO** (Rafailov et al., 2023) — direct preference optimization (DPO) without a reward model
- **Llama 3** (Grattafiori et al., 2024) and **Tulu 3** (Lambert et al., 2024) — modern multi-stage recipes
- **DeepSeek R1** (Guo et al., 2025) — popularized RLVR

Other prominent, early RLHF work on language models

- **Stiennon et al. 2020** (Stiennon et al., 2020) — Learning to summarize from human feedback. Extended Ziegler et al. to long-form summarization with PPO
- **WebGPT** (Nakano et al., 2021) — Trained a model to browse the web and answer questions using RLHF, with citations
- **Anthropic HH** (Bai et al., 2022) — Training a helpful and harmless assistant with RLHF. Introduced the “HH” dataset widely used in open research
- **GopherCite** (Menick et al., 2022) — Teaching language models to support answers with verified quotes (DeepMind)
- **Sparrow** (Glaese et al., 2022) — Improving alignment of dialogue agents via targeted human judgements (DeepMind). Added rule-based constraints to RLHF
- **InstructGPT** (Ouyang et al., 2022) — The canonical three-step RLHF recipe that powered ChatGPT

What RLHF comparison data looks like

A human (or AI) annotator sees two responses to the same prompt and picks the better one — this preference pair becomes training data for the reward model.

USER

Explain why the sky is blue in one sentence.

ASSISTANT (Response A ✓)

The sky is blue due to Rayleigh scattering, where shorter blue wavelengths of sunlight are scattered more by atmospheric molecules than longer wavelengths.

USER

Explain why the sky is blue in one sentence.

ASSISTANT (Response B)

The sky appears blue because of the way light interacts with the atmosphere and stuff, it's basically just physics.

Classical RL vs. RLHF

Classical RL

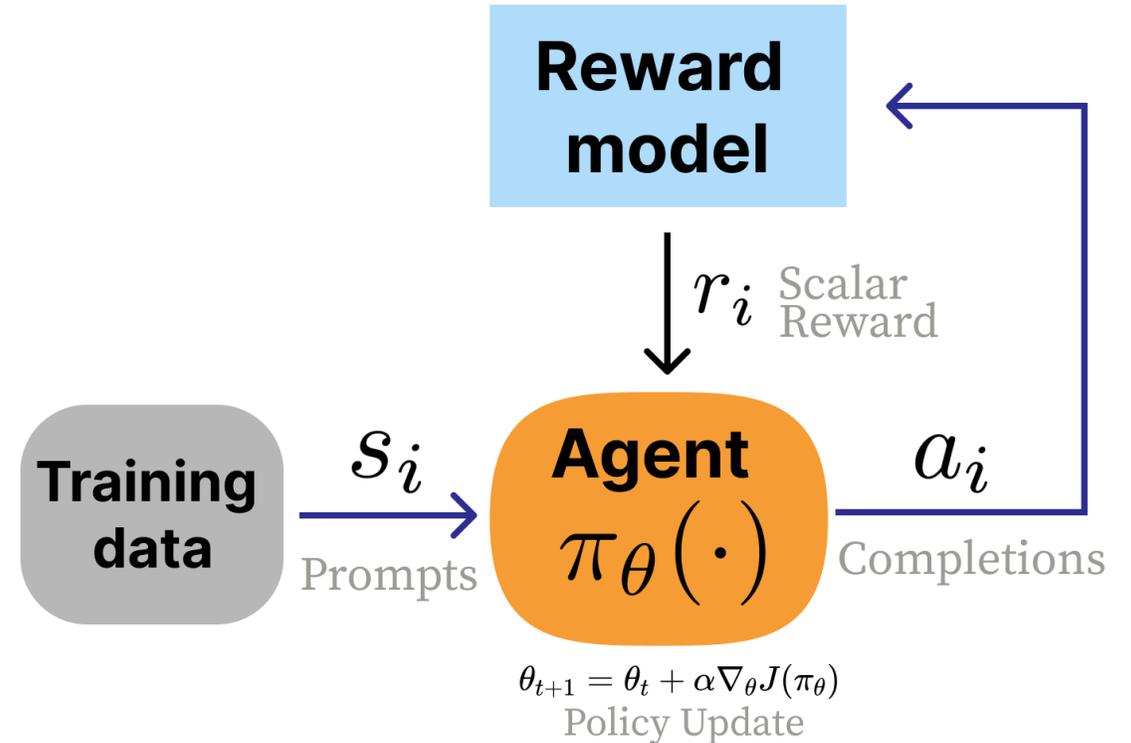
- Agent takes actions a_t in an environment with states s_t
- Reward is a known function $r(s_t, a_t)$ from the environment per step
- Optimize cumulative return over a trajectory (total steps T)

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \right]$$

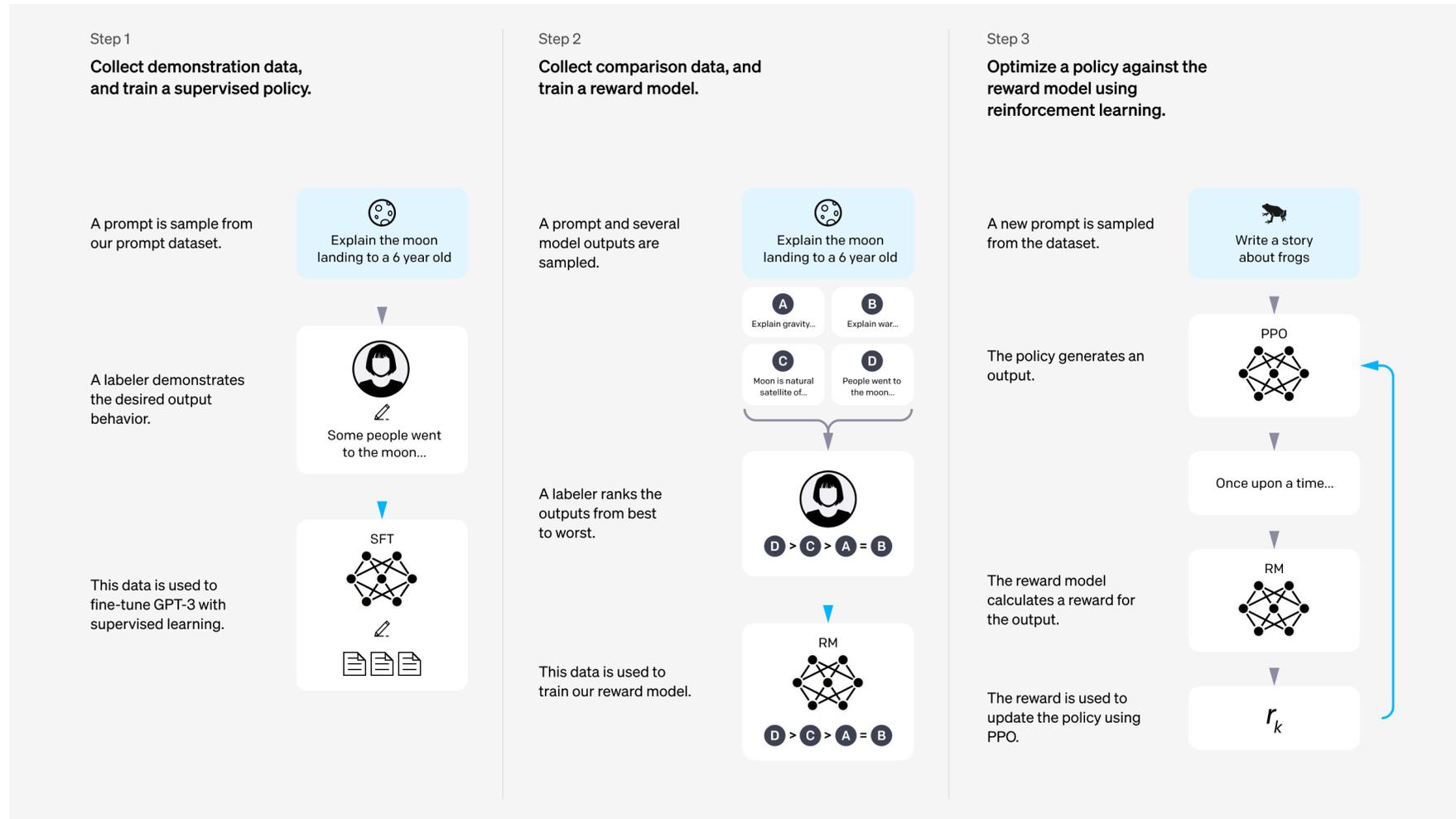
RLHF

- No environment — prompts sampled from a dataset
- Reward is **learned** from human preferences (a proxy)
- **Response-level** reward (bandit-style, not per-token)
- Regularized with **KL penalty** to stay close to the base model

$$J(\pi) = \mathbb{E} [r_\theta(x, y)] - \beta D_{\text{KL}}(\pi \parallel \pi_{\text{ref}})$$



InstructGPT's 3-step RLHF recipe



The 3-step RLHF process figure from InstructGPT, which became “the standard” approach to RLHF for a few years.

Step 1/3: Instruction fine-tuning (IFT)

The foundation of post-training. Also called **Supervised Fine-tuning (SFT)**:

- Start from a pretrained language model
- Collect demonstrations of *desired* assistant behavior
- Train with standard supervised learning on prompt-response pairs.
(different batch size, learning rate, etc.)
- Model can now answer questions.
Easy to use IFT to quickly adapt base model to many domains.

$$\mathcal{L}_{\text{SFT}}(\theta) = - \sum_{(x, y^*)} \sum_{t=1}^{|y^*|} \log \pi_{\theta}(y_t^* \mid x, y_{<t}^*)$$

SYSTEM

You are a helpful, harmless assistant. A system message like this can be used to steer the model to specific personas or behaviors.

USER

Write me a short poem about an optimistic goldfish.

ASSISTANT

Bright little goldfish
Finds a sunrise in each wave
Small bowl, endless hope

Step 2/3: Reward modeling

Overview:

- Collect **comparisons** between two model outputs for the same prompt
- RLHF gets its name from collecting *human* feedback between completions, but today much of it is AI feedback
- Train a reward model $r_\phi(x, y)$ to score preferred completions higher

The probability model says a response should win when it gets a higher reward score:

$$P(y_w \succ y_l \mid x) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))$$

Training then minimizes the negative log-likelihood of the preferred response beating the rejected one:

$$\mathcal{L}_{\text{RM}}(\phi) = -\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))$$

Notation:

- x is the prompt
- y_w is the **winning** response
- y_l is the **losing** response
- $r_\phi(x, y)$ is the trained reward model

Step 2/3: Reward modeling

Core Idea

The reward used in RLHF is the model predicting the probability that a given piece of text would be the "winning" or "chosen" completion in a pair/batch.

Clever!

The probability model says a response should win when it gets a higher reward score:

$$P(y_w \succ y_l \mid x) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))$$

Training then minimizes the negative log-likelihood of the preferred response beating the rejected one:

$$\mathcal{L}_{\text{RM}}(\phi) = -\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))$$

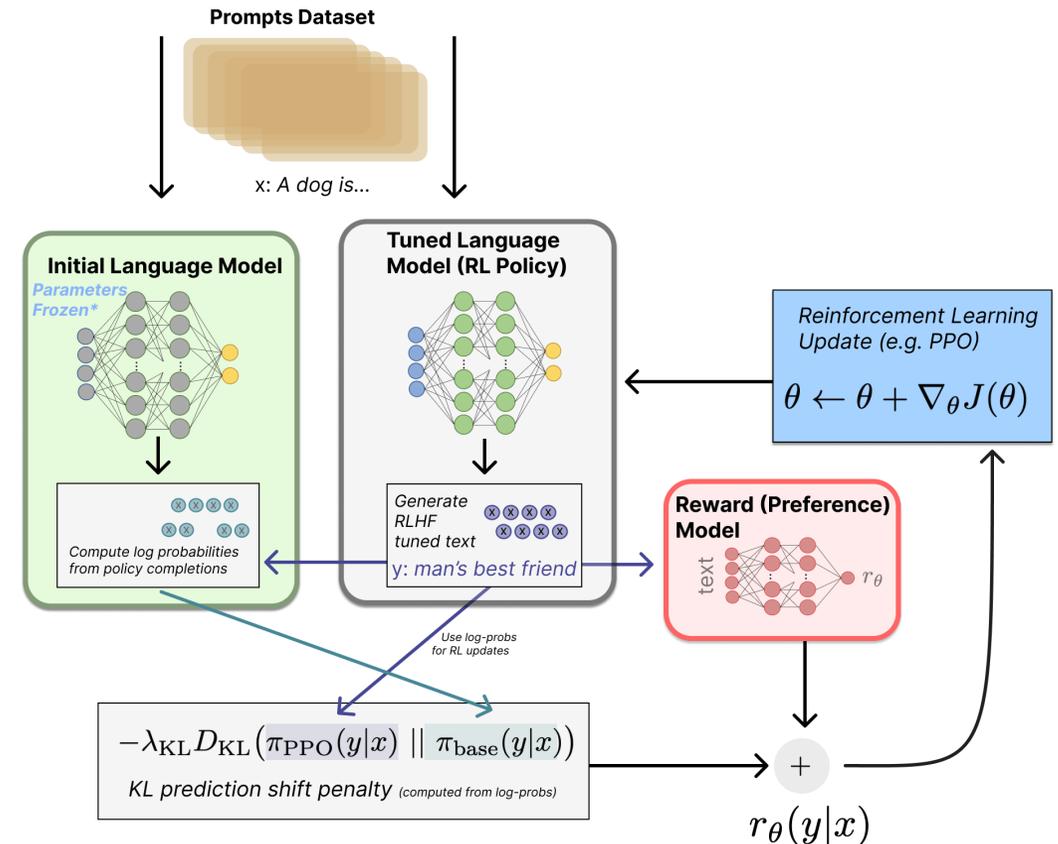
Notation:

- x is the prompt
- y_w is the **winning** response
- y_l is the **losing** response
- $r_\phi(x, y)$ is the trained reward model

Step 3/3: RL against the reward model

Where everything comes together (and RLHF gets its name):

- Sample a batch of prompts x_i from the dataset \mathcal{D}
- Generate completions $y_i \sim \pi_\theta(\cdot | x_i)$ from the model being trained
- Score them with the reward model $r_\phi(x_i, y_i)$
- Add a **KL penalty** so the policy stays close to the SFT/reference model.¹
- Update the policy with a policy-gradient RL algorithm (Proximal Policy Optimization, PPO in InstructGPT & ChatGPT)



Ouyang et al., 2022

¹: KL divergence measures how much the current policy differs from the reference model. For discrete outputs, $D_{\text{KL}}(\pi || \pi_{\text{ref}}) = \mathbb{E}_{y \sim \pi}[\log \pi(y | x) - \log \pi_{\text{ref}}(y | x)]$. People often colloquially call this the "KL distance" between the models, even though it is not a true metric.

The RLHF objective, unpacked

$$\max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi(\cdot | x)} \underbrace{r_{\phi}(x, y)}_{\text{maximize the reward}} - \underbrace{\beta D_{\text{KL}}(\pi(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x))}_{\text{but don't change the model too much}}$$

The reference model π_{ref} keeps the policy anchored to the SFT model.

$D_{\text{KL}}(\pi(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x))$ measures how far the new policy moves from that reference on prompt x .

β controls the tradeoff between **improving behavior** and **staying close** to what the model already knows.

What if we optimize this more directly?

$$\max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi(\cdot | x)} r_{\phi}(x, y) - \beta D_{\text{KL}}(\pi(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x))$$

Direct Preference Optimization (DPO)

- Derived the gradient toward the optimal solution, π^* , to the above equation
- Eliminated the need for a separate reward model (via training an implicit one)
- Train directly on preferred (y_w) vs. rejected (y_l) responses to a prompt (x)

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right)$$

What if we optimize this more directly?

$$\max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi(\cdot | x)} r_{\phi}(x, y) - \beta D_{\text{KL}}(\pi(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x))$$

Direct Preference Optimization (DPO)

- Derived the gradient toward the optimal solution, π^* , to the above equation
- Eliminated the need for a separate reward model (via training an implicit one)
- Train directly on preferred (y_w) vs. rejected (y_l) responses to a prompt (x)

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right)$$

DPO became very popular as it is

- Far simpler to implement
- Far cheaper to run
- Achieves ~80% or more of the final performance
- I used it to build models like Zephyr-Beta, Tulu 2/3, Olmo 2/3, etc.

Rejection sampling: the simplest preference optimization

Generate many completions, score them, fine-tune on the best:

1. **Generate:** sample N completions per prompt from the current model
2. **Score:** pass each through the reward model $r_\phi(x, y)$
3. **Select:** keep the highest-scoring completion(s)
4. **Fine-tune:** standard SFT loss on the curated set

$$\mathcal{L}_{\text{RS}} = \mathcal{L}_{\text{SFT}}(\theta; \{(x_i, y_i^*)\}) \quad \text{where } y_i^* = \arg \max_j r_\phi(x_i, y_{i,j})$$

Simple, stable, and widely used: Llama 2 (Touvron et al., 2023) and DeepSeek R1 (Guo et al., 2025) both include rejection sampling stages.

The preference tuning landscape

	Rejection Sampling	Online RL (PPO)	DPO
Mechanism	Filter, then SFT	Generate, score, update policy	Direct gradient on preferences
Reward model	Required	Required	Implicit (no separate RM)
On-policy data	Yes (generate from current model)	Yes (generate each step)	No (fixed preference dataset)
Complexity	Low	High	Low

All three optimize the same underlying objective — they differ in **how** they move the policy toward higher-reward completions. There is substantial debate on which of these is the best for final performance, which RL generally wins, but evidence is mixed.

Caveat: proxy objectives and over-optimization

The reward model is a **proxy**, not ground truth. Even a well-trained RM is only *correlated* with real user satisfaction.

Goodhart's Law: “When a measure becomes a target, it ceases to be a good measure.”

What this looks like in practice:

- **Reward hacking:** RM score keeps climbing, but actual quality degrades
- **Verbosity bias:** longer responses score higher, so models become verbose
- **Sycophancy:** model tells users what they want to hear rather than being accurate
- **Over-refusal:** model refuses legitimate queries (e.g. “how to kill a linux process”)

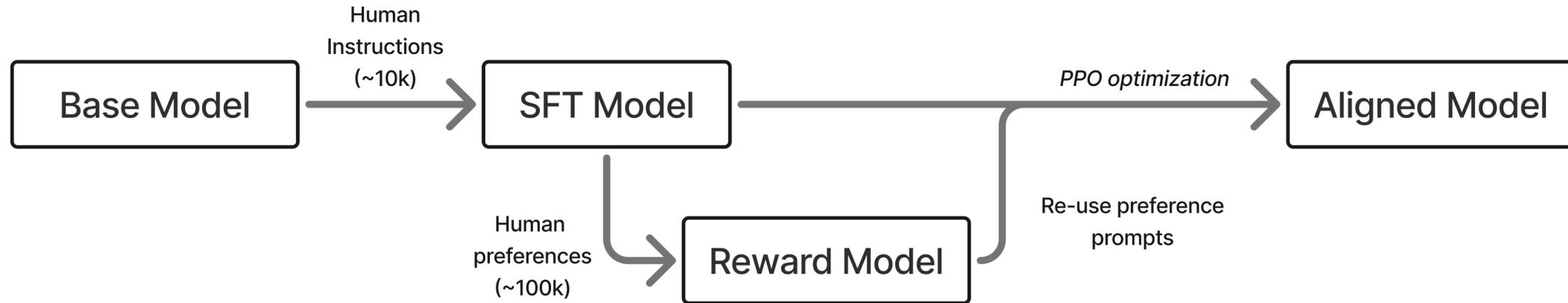
The KL penalty β is the main defense — it limits how far the policy can drift from the reference model. But over-optimization is a **fundamental tension** in all preference-based training.

How training recipes have evolved

	InstructGPT (2022)	Tülu 3 (2024)	DeepSeek R1 (2025)
Instruction data	~10K	~1M	100K+
Preference data	~100K	~1M	On-policy
RL stage	~100K prompts	~10K (RLVR)	N/A

An overall trend is to use far more compute across all the stages, but shifting more to RLVR.

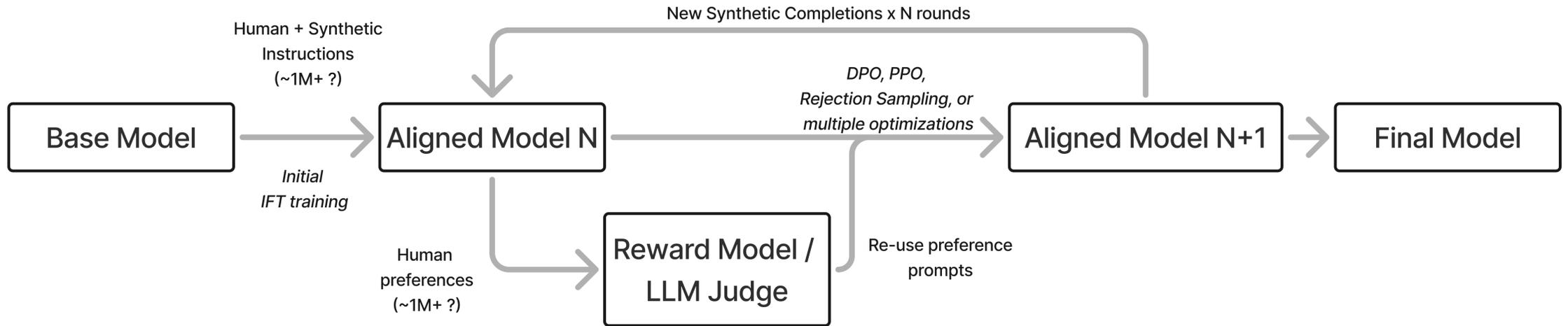
The early days: InstructGPT



Early on, RLHF had a well-documented, simple enough approach.

- **InstructGPT** made the classic three-stage recipe canonical: SFT, reward modeling, then RL against the reward model. *OpenAI even hinted that the original ChatGPT used this!*
- This became the intellectual template for much of modern post-training.

From RLHF to post-training



What began as an “RLHF” recipe evolved into a complex series of steps to get the final, best model (e.g. Nemotron 4 340B, Llama 3.1).

- Modern systems keep the same core idea of using multiple optimizers with different strengths and weaknesses, but add more stages, more data, and more filtering.
- This trend has only continued, and recipes ebb and flow, as tools like RLVR and model merging change the scope of what is doable in different ways.

From RLHF to “post-training”

As time has passed since ChatGPT, the field has gone through multiple distinct phases (roughly):

1. 2023: Simple SFT for better chatbots and reproducing RLHF fundamentals (Alpaca, Vicuna, etc.)
2. 2024: DPO dominates open models and training stages expand (Zephyr-beta, Tülu 2, etc.)
3. 2025: RLVR, complex recipes (Tülu 3, Olmo 3, Nemotron 3, R1, etc.)
4. 2026: Agentic training, multi-turn RL, etc.

From RLHF to “post-training”

As time has passed since ChatGPT, the field has gone through multiple distinct phases (roughly):

1. 2023: Simple SFT for better chatbots and reproducing RLHF fundamentals (Alpaca, Vicuna, etc.)
2. **2024: DPO dominates open models and training stages expand** (Zephyr-beta, Tülu 2, etc.)
3. 2025: RLVR, complex recipes (Tülu 3, Olmo 3, Nemotron 3, R1, etc.)
4. 2026: Agentic training, multi-turn RL, etc.

Within 2024 the field shifted its focus to post-training, as training stages evolved beyond the InstructGPT-style recipe, DPO proliferated, and largely RLHF was viewed as one tool (that you may not even need).

“Just style transfer”

RLHF’s reputation was that its contributions are minor on the final language models.

“A model’s knowledge and capabilities are learnt almost entirely during pretraining, while alignment teaches it which subdistribution of formats should be used when interacting with users.”

LIMA: Less Is More for Alignment (2023)

“Just style transfer”

RLHF’s reputation was that its contributions are minor on the final language models.

“A model’s knowledge and capabilities are learnt almost entirely during pretraining, while alignment teaches it which subdistribution of formats should be used when interacting with users.”

LIMA: Less Is More for Alignment (2023)

Sometimes this view of alignment (or RLHF) teaching “format” made people think that post-training only made minor changes to the model. This would describe finetuning as “*just style transfer.*”

The base model trained on trillions of tokens of web text has seen and learned from an extremely broad set of examples. The model at this stage contains far more latent capability than early post-training recipes were able to expose.

The question is: How does post-training interact with these?

An intuition for post-training

RLHF's reputation was that its contributions are minor on the final language models.

An example, **OLMoE** — same base model family, updated only post-training:

- `OLMoE-1B-7B-0924-Instruct` (Sep. 2024):
38.44 avg. eval score
- `OLMoE-1B-7B-0125-Instruct` (Jan. 2025):
45.62 avg. eval score

Base models determine the *ceiling*. Post-training's job has been to **reach it**.

Simple post-training often doesn't extract nearly enough performance (especially when the pace of progress is high).

An intuition for post-training

RLHF's reputation was that its contributions are minor on the final language models.

“A model’s knowledge and capabilities are learnt almost entirely during pretraining, while alignment teaches it which subdistribution of formats should be used when interacting with users.”

LIMA: Less Is More for Alignment (2023)

“The superficial alignment hypothesis (SAH) posits that large language models learn most of their knowledge during pre-training, and that post-training merely surfaces this knowledge.”

Operationalising the Superficial Alignment Hypothesis via Task Complexity (2026)

The second paper, 3 years later, matches my intuition for post-training.

I call this the Elicitation Theory of post-training, where we're trying to pull out the most useful knowledge of the model.

An intuition for post-training

RLHF's reputation was that its contributions are minor on the final language models.

“A model’s knowledge and capabilities are learnt almost entirely during pretraining, while alignment teaches it which subdistribution of formats should be used when interacting with users.”

LIMA: Less Is More for Alignment (2023)

“The superficial alignment hypothesis (SAH) posits that large language models learn most of their knowledge during pre-training, and that post-training merely surfaces this knowledge.”

Operationalising the Superficial Alignment Hypothesis via Task Complexity (2026)

The second paper, 3 years later, matches my intuition for post-training.

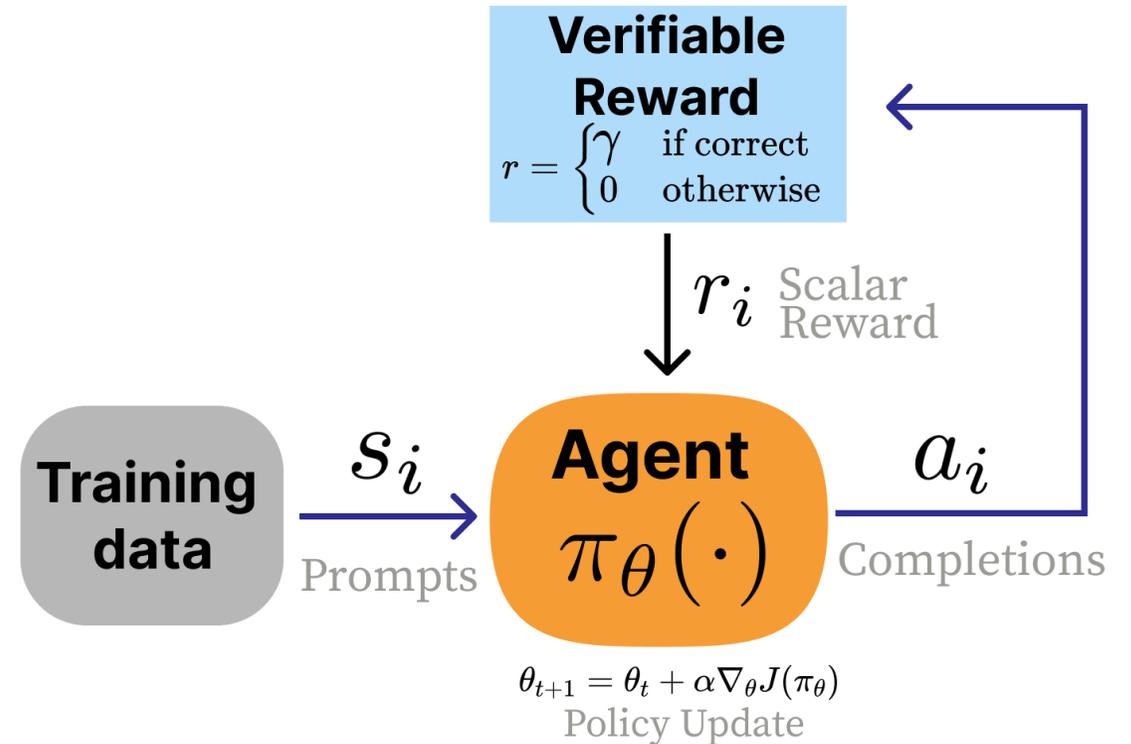
I call this the Elicitation Theory of post-training, where we're trying to pull out the most useful knowledge of the model.

Pretraining builds a chassis for the car -- post-training is the hard craft of extracting the most performance from it.

Reinforcement learning with *Verifiable* rewards

Apply the same RL algorithms to LLMs when the answer can be checked directly. No need to train a reward model:

- E.g. Math: check the final answer. Code: run the tests.
- No learned reward model — **no proxy objective**
- Enables scaling RL compute on reasoning tasks
- Unlocked **inference time scaling**: Spending more compute at generation time per problem increases performance log-linearly w.r.t. compute
- RLVR was named by **Tülu 3** (Lambert et al., 2024) and popularized by **DeepSeek R1** (Guo et al., 2025)

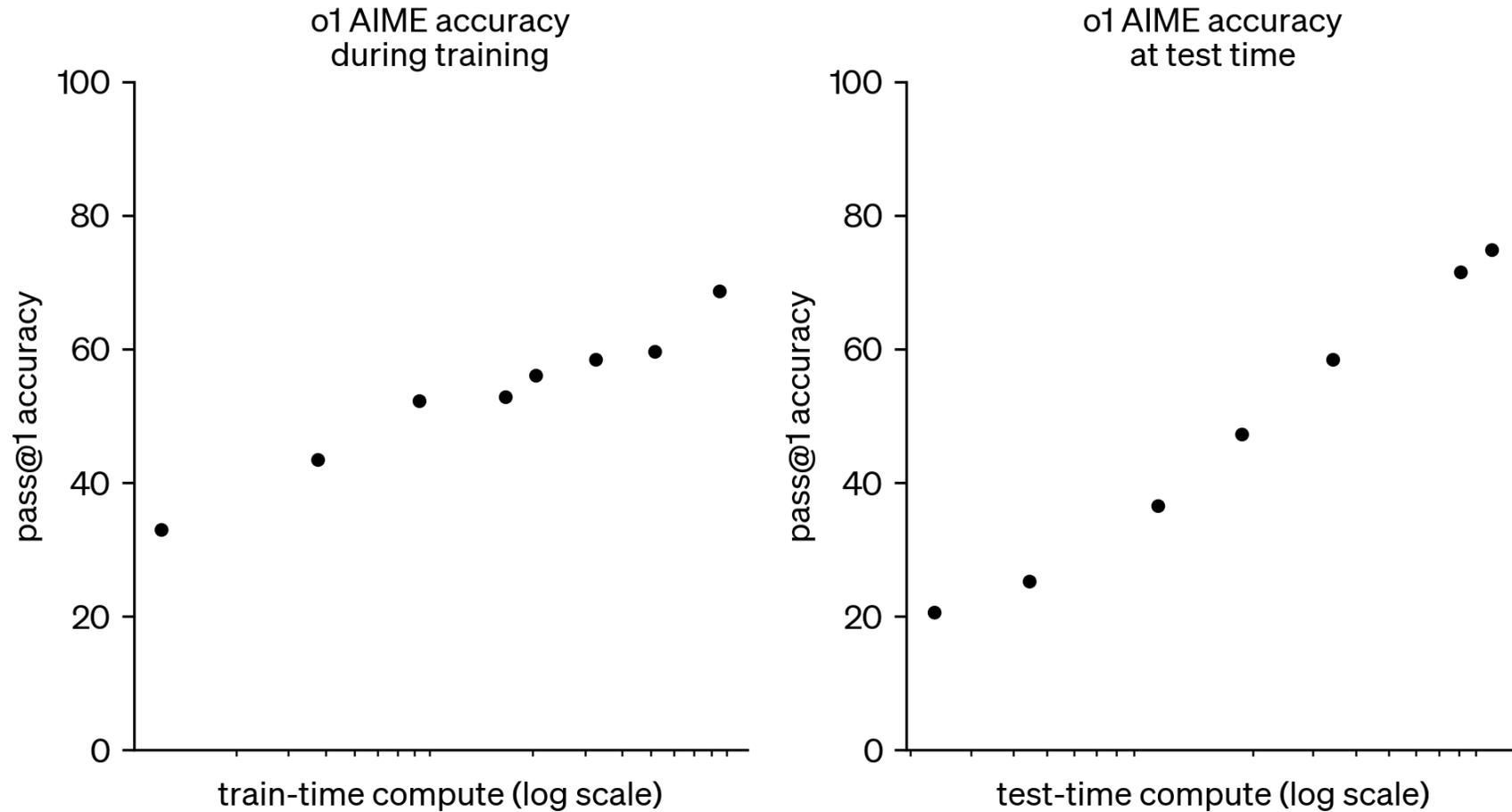


Comparing classical RL vs. LLM RLHF and RLVR

	Classical RL	RLHF	RLVR
Reward	Environment	Learned (proxy)	Verifiable (exact)
State transitions	Yes	No	No
Reward granularity	Per-step	Per-response	Per-response
Primary challenge	Explore-Exploit Trade-off	Over-optimization	Task generalization
Example	CartPole	Chat style tuning	Math reasoning

Beyond elicitation: The scaling RL era of post-training

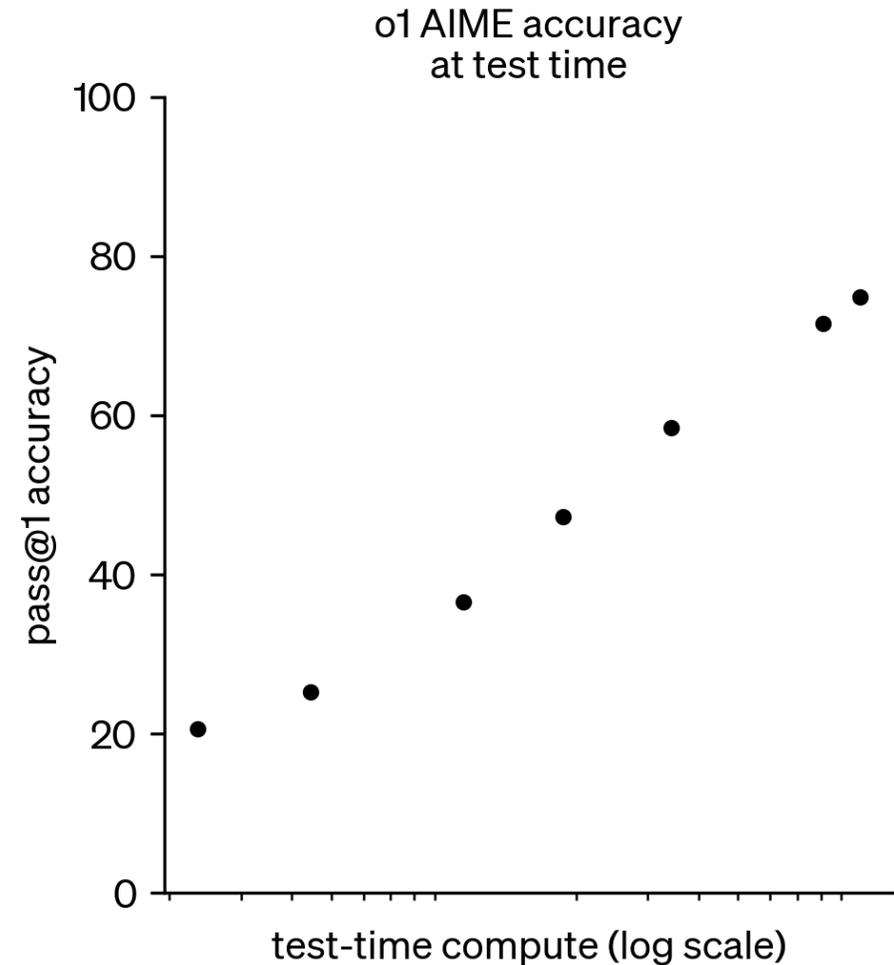
OpenAI's seminal scaling plot with o1-preview



o1: Test-time scaling

A log-linear relationship between inference compute (number of tokens generated) and downstream performance.

- This is a fundamental property of models, unlocked in its popular form with RLVR
- Can be done in many ways: One long chain of thought (CoT) sequence, multiple agents in parallel, or mixes of the two
- Improving inference-time scaling changes the slope and offset of the curve

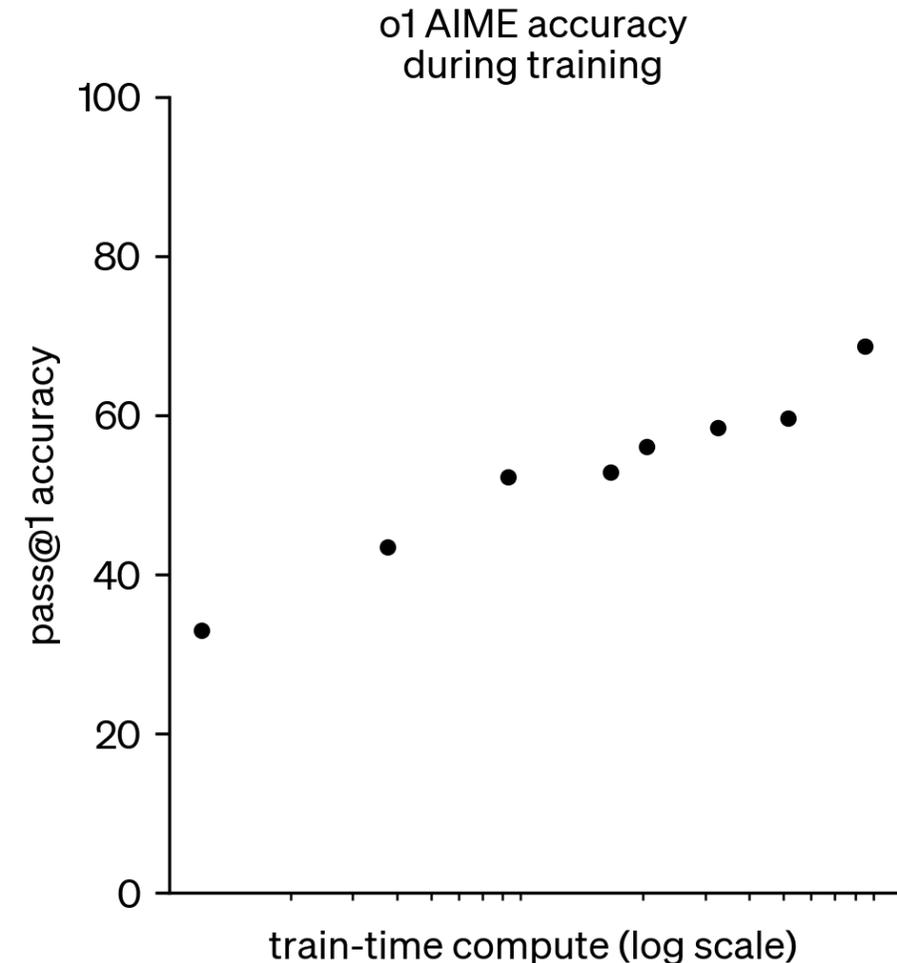


o1: Training-time scaling (with reinforcement learning!)

An often underplayed portion of the o1 release (and future reasoning/agentic models).

- Scaling reinforcement learning compute also has a log-linear return on performance!
- The core question: Is scaling RL *training* just eliciting more from the base model or actually teaching new abilities?

Results in a two-sided scaling landscape for training language models – both pretraining and post-training. The third place of scaling is at inference (no weight updates there).



Cursor Composer 1.5: RL scaling

Composer 1.5 scales reinforcement learning
20x beyond the original model

Cursor Bench

60

50

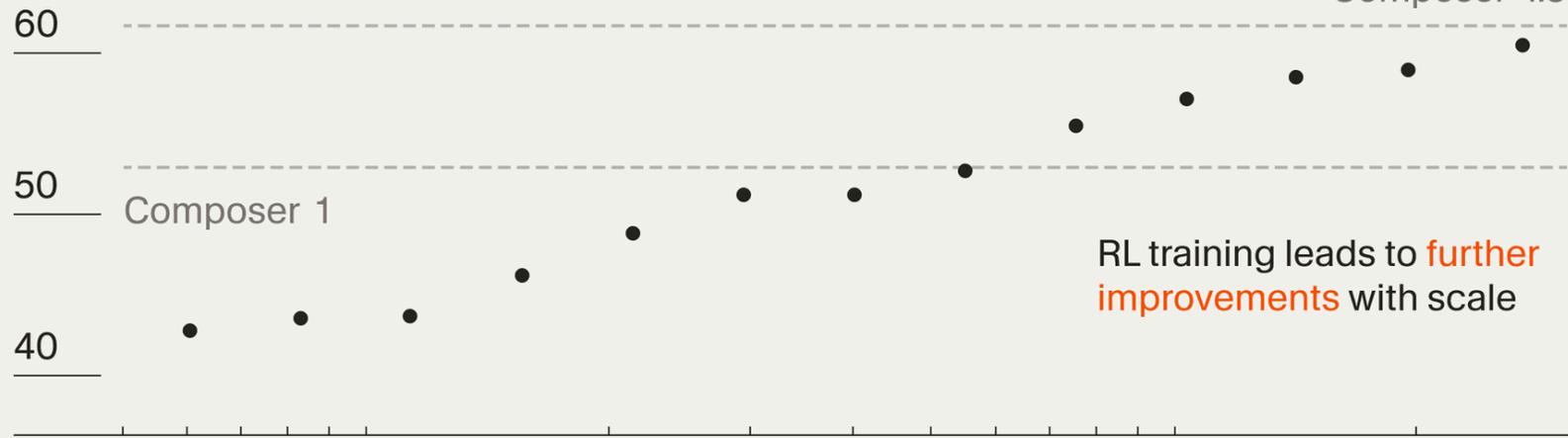
40

Composer 1

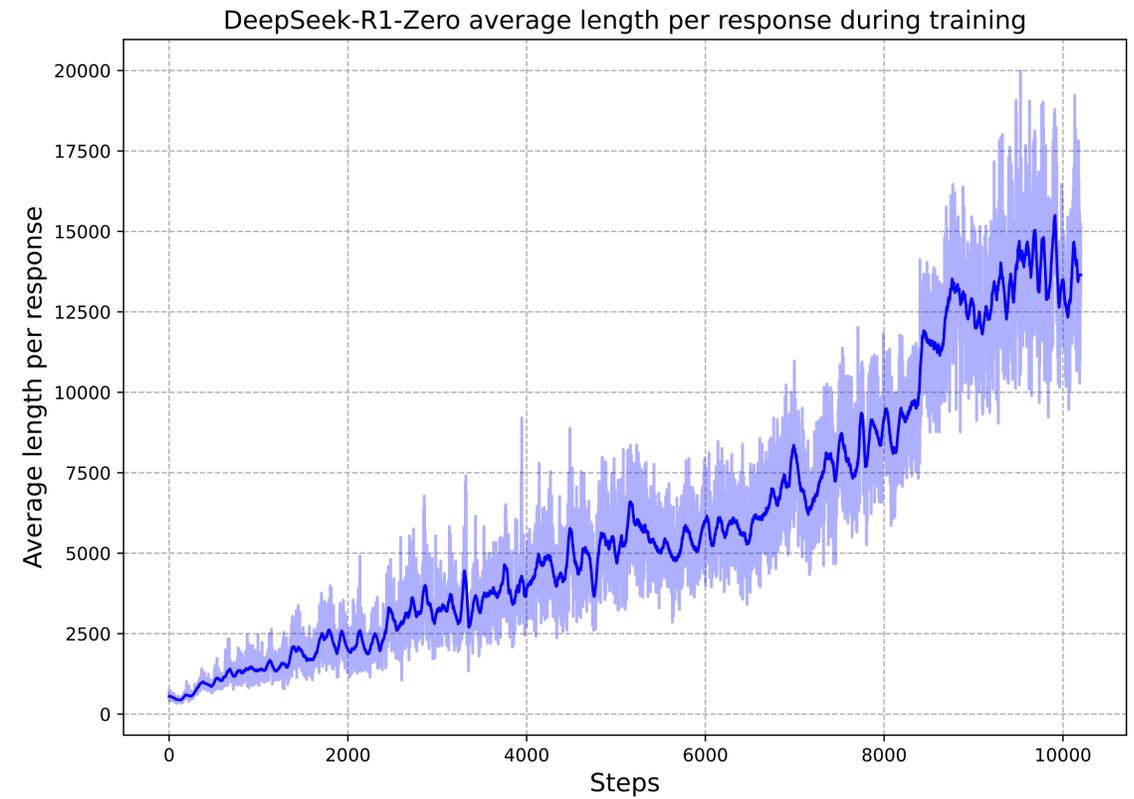
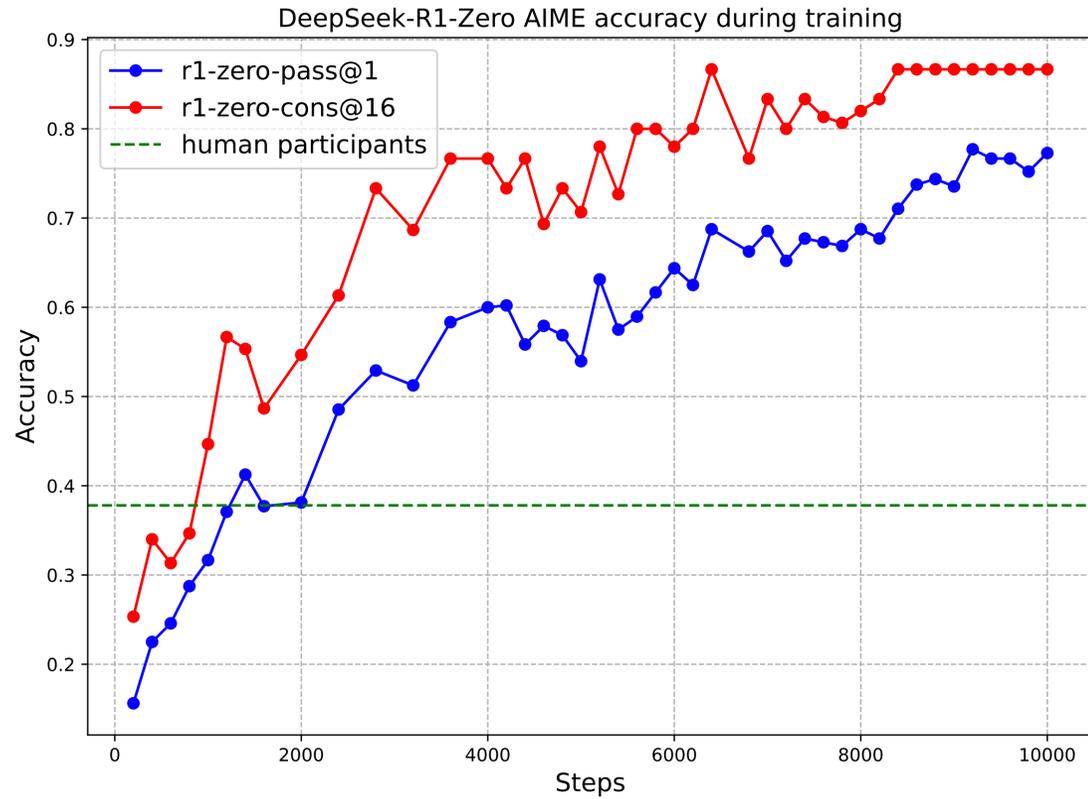
Composer 1.5

RL training leads to **further improvements** with scale

Training compute (log)



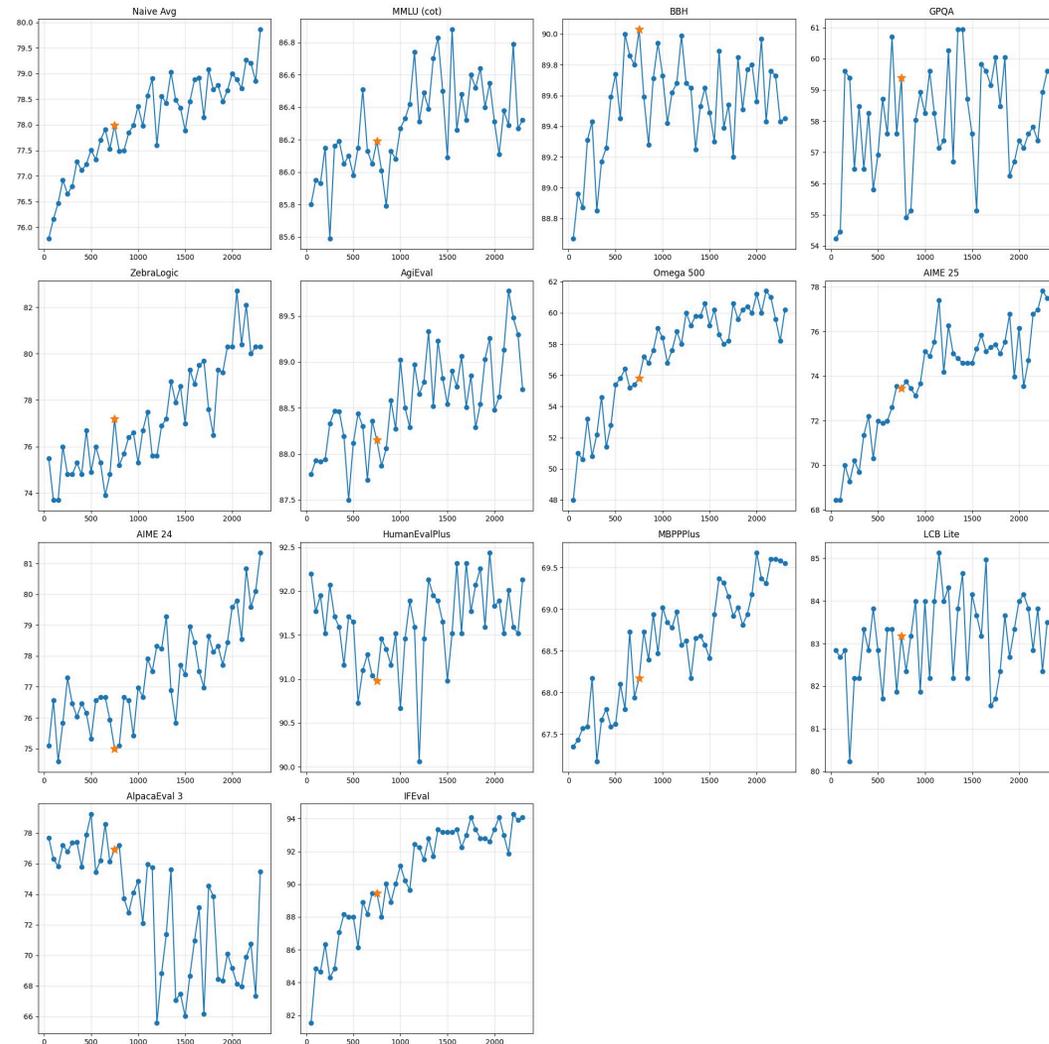
DeepSeek-R1-Zero: RL scaling



Olmo 3.1: extending the RL run

One of the few “fully open” large-scale RL runs to date.

- Training a general, 32B reasoning model.
- Full RL training took about **28 days on 224 GPUs**.
- Improvements in performance were very consistent across the run, in fact they were still going up when we had to stop it!



Where this leaves us

Post-training and RLHF are changing faster than maybe ever before.

- Language models are becoming “tool-use native” and are now about tools, harnesses (how you tell the model to use said tools), and much more than just weights
- RLHF and human preferences haven’t gone away, but are evolving far more slowly and out of the central gaze of the industry
- Building language models and doing research is changing rapidly with coding agents

Lecture 1: Overview

Lecture 1

1. Introduction
2. Key Related Works
3. Training Overview

Core Training Pipeline

4. Instruction Tuning
5. Reward Models
6. Reinforcement Learning
7. Reasoning
8. Direct Alignment
9. Rejection Sampling

Data & Preferences

10. What are Preferences
11. Preference Data
12. Synthetic Data & CAI

Practical Considerations

13. Tool Use
14. Over-optimization
15. Regularization
16. Evaluation
17. Product & Character

Appendices

- A. Definitions
- B. Style & Information
- C. Practical Issues

Course Home

- rlhfbook.com
- Shared references and future lecture decks live here

Thank you

Questions / discussion

Contact: nathan@natolambert.com

Newsletter: interconnects.ai

rlhfbook.com



BUILT WITH
[natolambert/colloquium](https://github.com/natolambert/colloquium)

References (1/5)

Anthropic. “*Claude Code*.” 2025. [link]

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., et al.. “*Constitutional ai: Harmlessness from ai feedback*.” *arXiv preprint arXiv:2212.08073*, 2022.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., et al.. “*Training a helpful and harmless assistant with reinforcement learning from human feedback*.” *arXiv preprint arXiv:2204.05862*, 2022.

Bender, E., Gebru, T., McMillan-Major, A., and Shmitchell, S.. “*On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?*” *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021. [link]

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., et al.. “*Language Models are Few-Shot Learners*.” *Advances in Neural Information Processing Systems*, 2020. [link]

Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S., et al.. “*Deep Reinforcement Learning from Human Preferences*.” *Advances in Neural Information Processing Systems*, 2017.

Devlin, J., Chang, M., Lee, K., and Toutanova, K.. “*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.” *arXiv preprint arXiv:1810.04805*, 2018. [link]

Gao, L., Schulman, J., and Hilton, J.. “*Scaling laws for reward model overoptimization*.” *International Conference on Machine Learning*, 2023.

References (2/5)

- Glaese, A., McAleese, N., Tr{\c{e}}bacz, M., Aslanides, J., Firoiu, V., et al.. “*Improving alignment of dialogue agents via targeted human judgements.*” *arXiv preprint arXiv:2209.14375*, 2022.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., et al.. “*The Llama 3 herd of models.*” *arXiv preprint arXiv:2407.21783*, 2024.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., et al.. “*Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.*” *arXiv preprint arXiv:2501.12948*, 2025.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., et al.. “*Scaling Laws for Neural Language Models.*” *arXiv preprint arXiv:2001.08361*, 2020. [\[link\]](#)
- Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., et al.. “*Llama 3: Pushing Frontiers in Open Language Model Post-Training.*” *arXiv preprint arXiv:2411.15124*, 2024.
- Menick, J., Trebacz, M., Mikulik, V., Aslanides, J., Song, F., et al.. “*Teaching language models to support answers with verified quotes.*” *arXiv preprint arXiv:2203.11147*, 2022.
- Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., et al.. “*OLMoE: Open Mixture-of-Experts Language Models.*” *International Conference on Learning Representations*, 2025. [\[link\]](#)
- Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., et al.. “*WebGPT: Browser-assisted question-answering with human feedback.*” *arXiv preprint arXiv:2112.09332*, 2021.

References (3/5)

Olmo, T., Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., et al.. “*Olmo 3.*” 2025. [\[link\]](#)

OpenAI. “*ChatGPT: Optimizing Language Models for Dialogue.*” *OpenAI Blog*, 2022. [\[link\]](#)

OpenAI. “*GPT-4 Technical Report.*” *arXiv preprint arXiv:2303.08774*, 2023. [\[link\]](#)

OpenAI. “*Introducing OpenAI o1-preview.*” *OpenAI Blog*, 2024. [\[link\]](#)

OpenAI. “*Introducing OpenAI o3 and o4-mini.*” *OpenAI Blog*, 2025. [\[link\]](#)

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., et al.. “*Training language models to follow instructions with human feedback.*” *Advances in Neural Information Processing Systems*, 2022.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., et al.. “*Deep Contextualized Word Representations.*” *Proceedings of NAACL-HLT*, 2018. [\[link\]](#)

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I.. “*Improving Language Understanding by Generative Pre-Training.*” 2018. [\[link\]](#)

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., et al.. “*Language Models are Unsupervised Multitask Learners.*” 2019. [\[link\]](#)

Rafailov, R., Sharma, A., Mitchell, E., Manning, C., Ermon, S., et al.. “*Direct preference optimization: Your language model is secretly a reward model.*” *Advances in Neural Information Processing Systems*, 2023.

References (4/5)

- Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., et al.. “*Learning to summarize from human feedback.*” *Advances in Neural Information Processing Systems*, 2020.
- Sutton, R., and Barto, A.. “*Reinforcement Learning: An Introduction.*” MIT Press, 2018. [\[link\]](#)
- Team, O.. “*OLMoE, meet iOS.*” *Ai2 Blog*, 2025. [\[link\]](#)
- Team, O.. “*New Tools for Building Agents.*” *OpenAI Blog*, 2025. [\[link\]](#)
- Team, C.. “*Introducing Composer 1.5.*” *Cursor Blog*, 2026. [\[link\]](#)
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., et al.. “*Llama 2: Open foundation and fine-tuned chat models.*” *arXiv preprint arXiv:2307.09288*, 2023.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al.. “*Attention Is All You Need.*” *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017. [\[link\]](#)
- Vergara-Browne, T., Patil, D., Titov, I., Reddy, S., Pimentel, T., et al.. “*Operationalising the Superficial Alignment Hypothesis via Task Complexity.*” *arXiv preprint arXiv:2602.15829*, 2026. [\[link\]](#)
- Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., et al.. “*LIMA: Less Is More for Alignment.*” *Advances in Neural Information Processing Systems*, 2023.

References (5/5)

Ziegler, D., Stiennon, N., Wu, J., Brown, T., Radford, A., et al.. “*Fine-tuning language models from human preferences.*” *arXiv preprint arXiv:1909.08593*, 2019.