Natural Language Processing

Scaling Axes toward the Frontier of Agentic Intelligence

Daniel (Joongwon) Kim

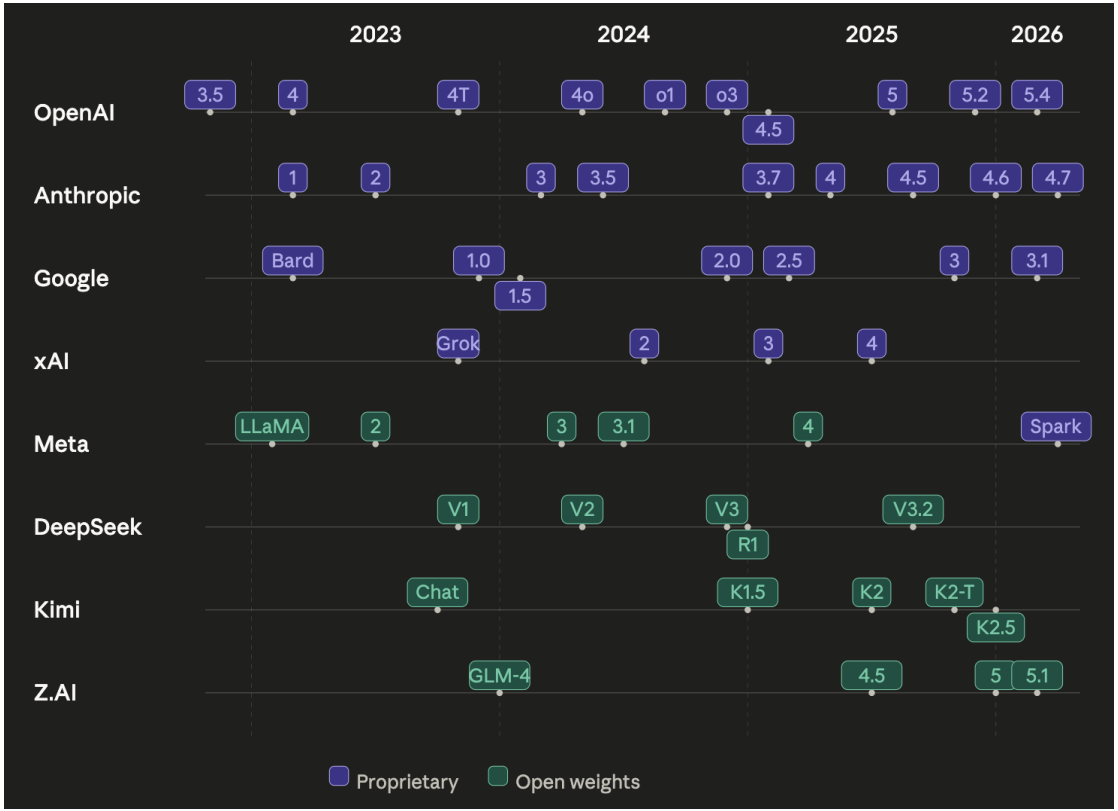
jwonkim@cs.washington.edu

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

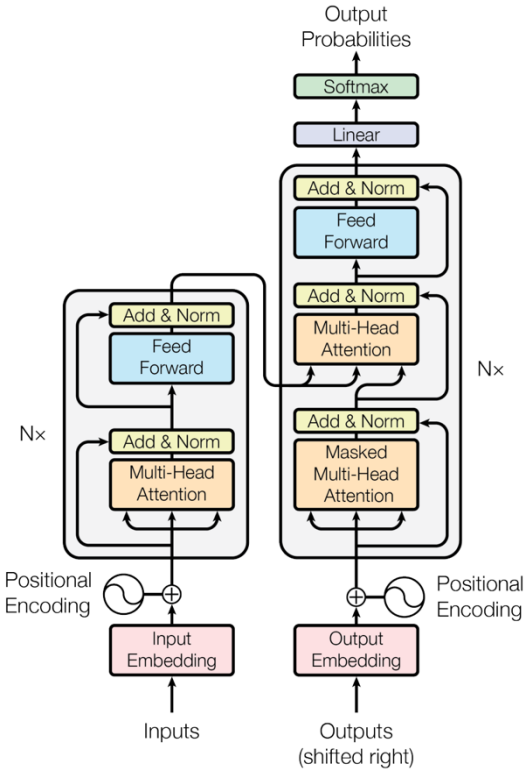
TA intro

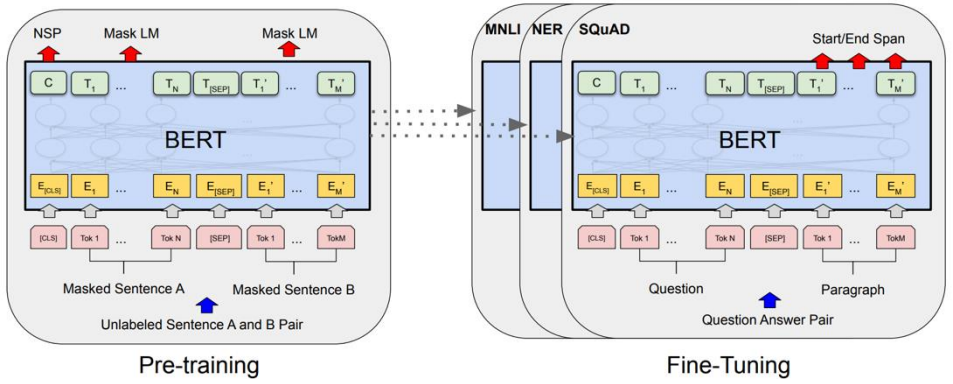
- Daniel Kim
- 4th-year CSE PhD student advised by Prof. Hanna Hajishirzi
- Working jointly at Meta (FAIR / Meta Superintelligence Labs)
- Research areas: reasoning, agents, post-training, test-time scaling

Evolution of LLMs (2022 → now)

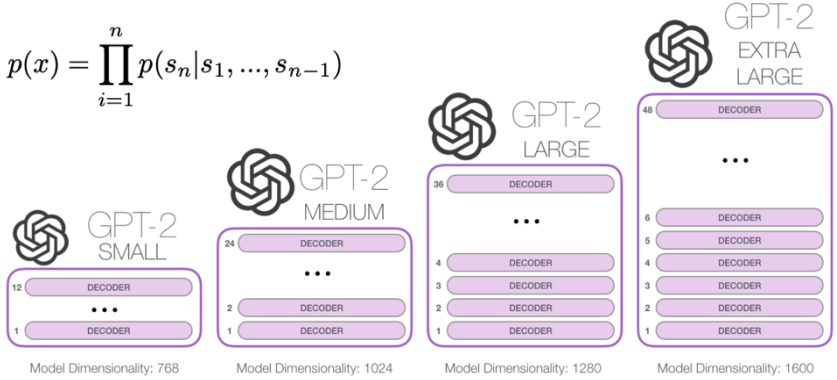


Early Years of Language Modeling (2017 – 19)

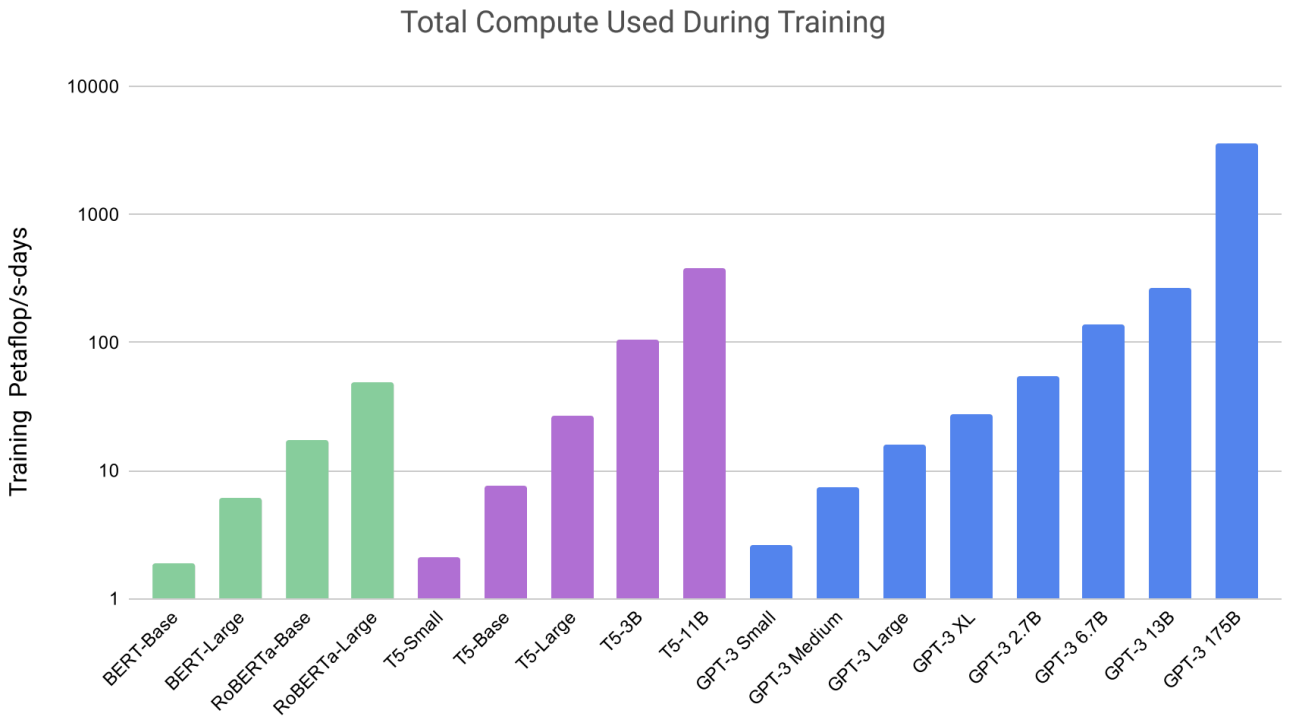




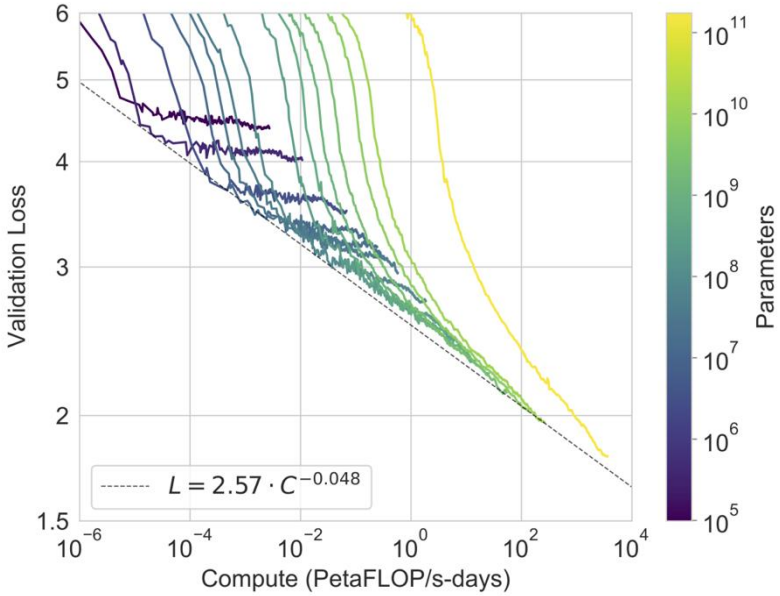
$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, ..., s_{n-1})$$

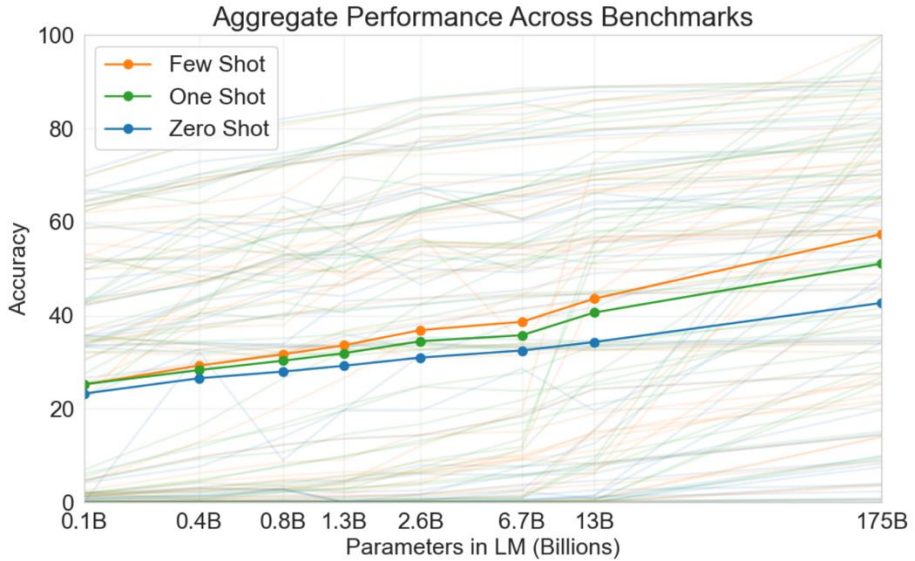


Scaling Laws with GPT-3



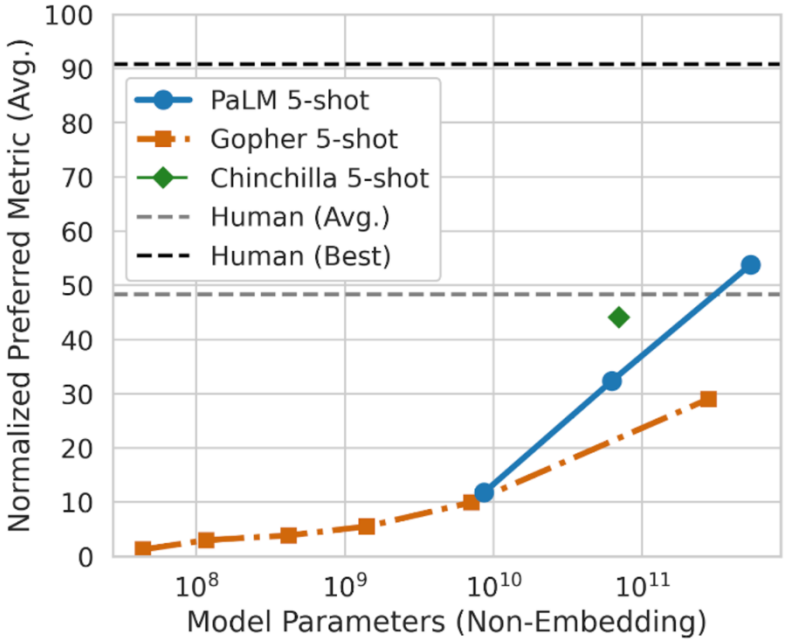
Scaling Laws with GPT-3

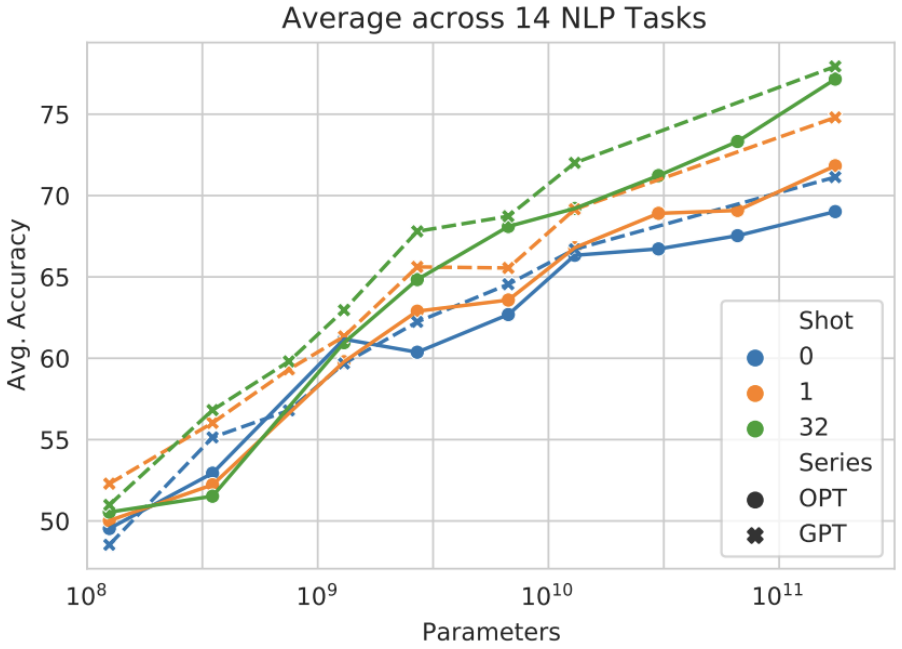




Language Models are Few-Shot Learners (OpenAI, 2020)

Scaling Laws with PaLM / OPT

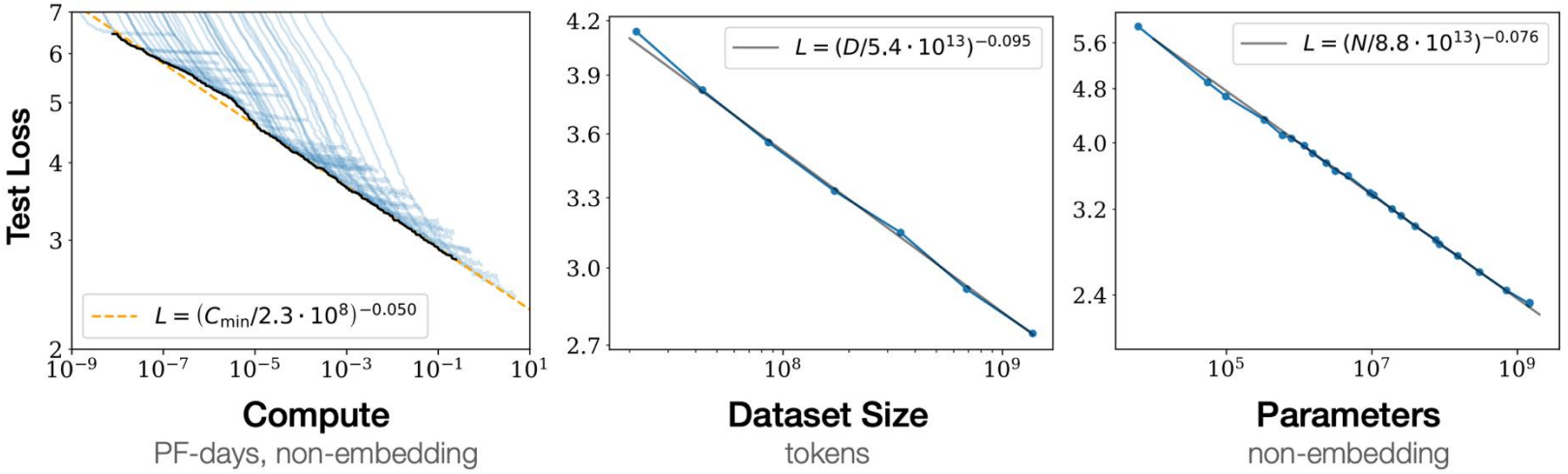




PaLM: Scaling Language Modeling with Pathways (Google, 2022)

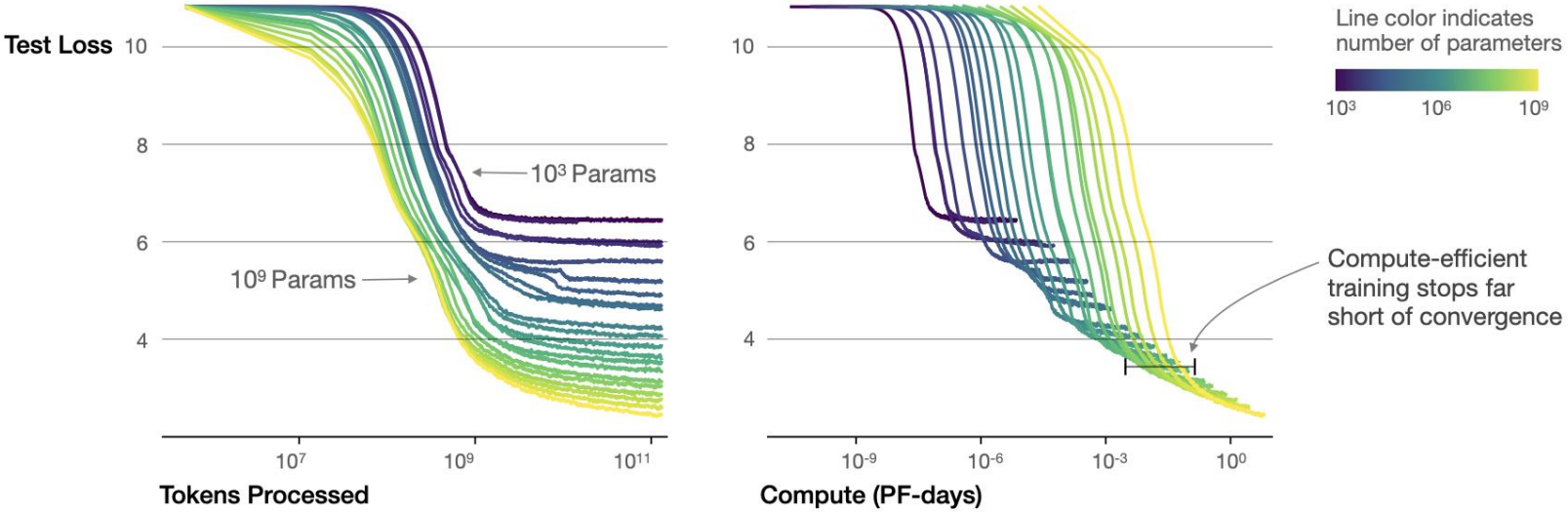
OPT: Open Pre-trained Transformer Language Models (Meta, 2022)

Scaling Laws (OpenAI)



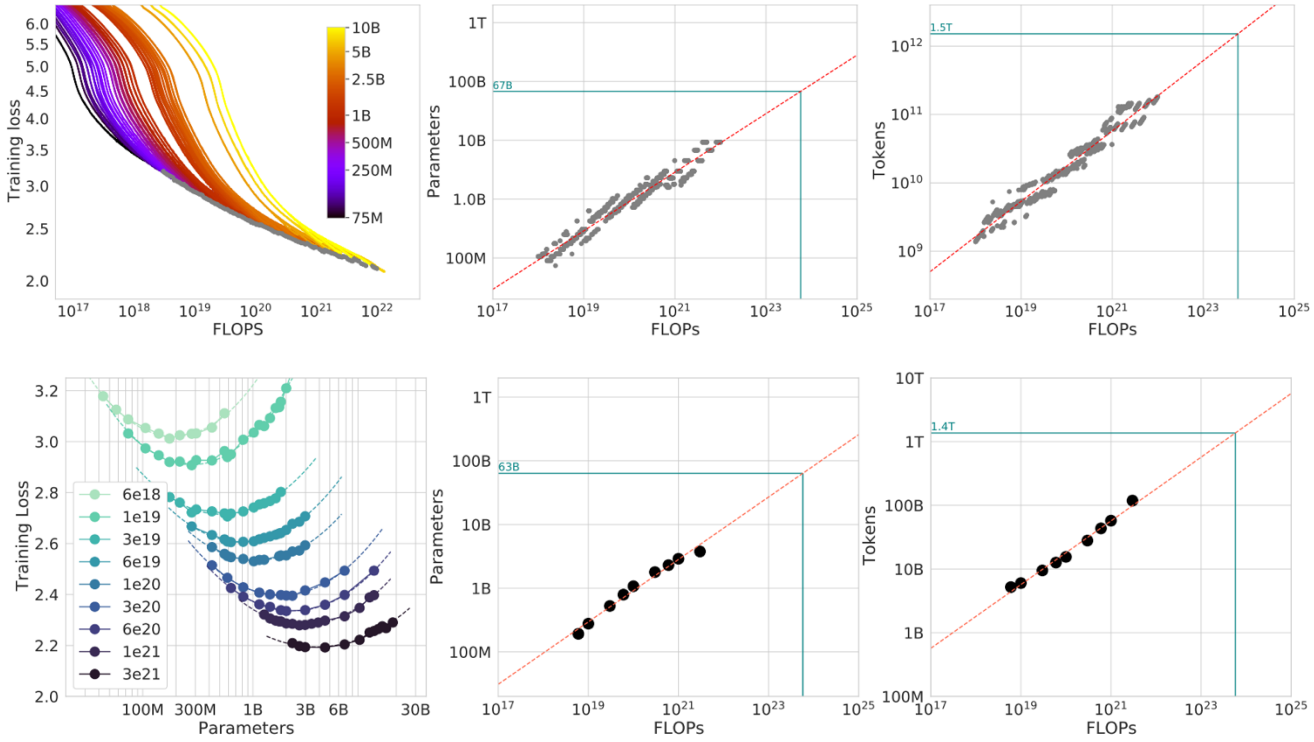
Scaling Laws for Neural Language Models (OpenAI, 2020)

Scaling Laws (OpenAI)



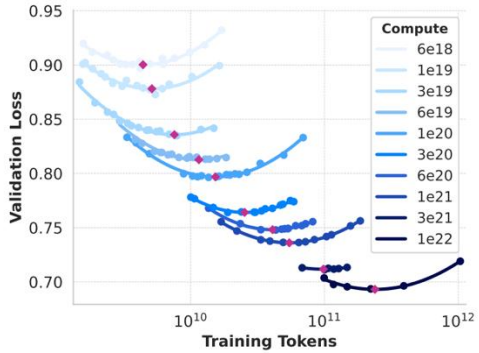
Scaling Laws for Neural Language Models (OpenAI, 2020)

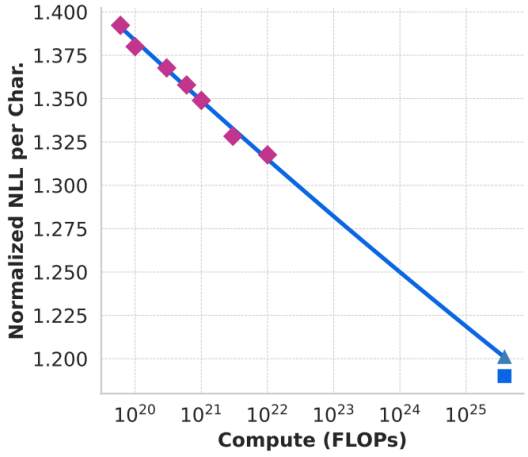
Scaling Laws (DeepMind)

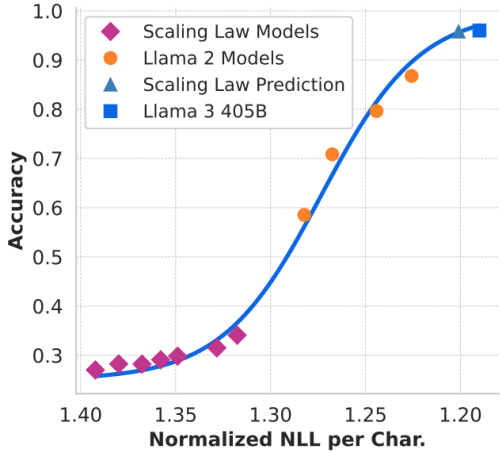


Training Compute-Optimal Large Language Models (DeepMind, 2022)

Scaling Laws (Meta)

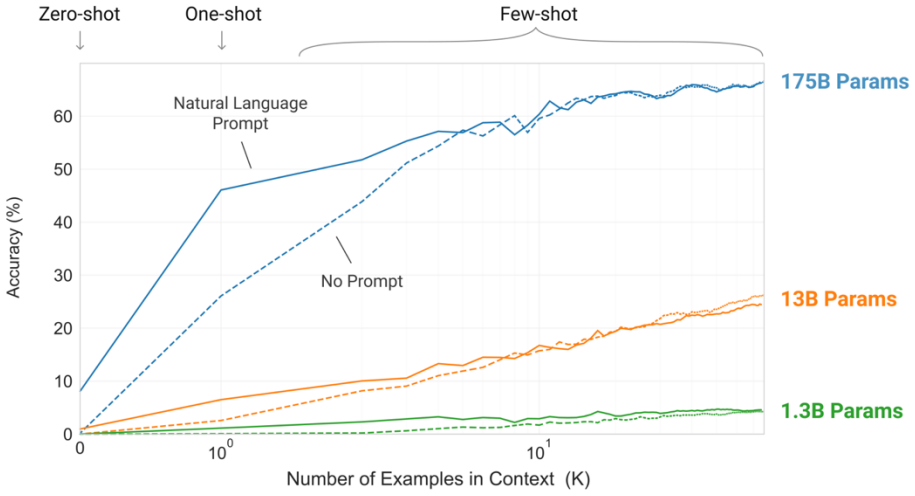


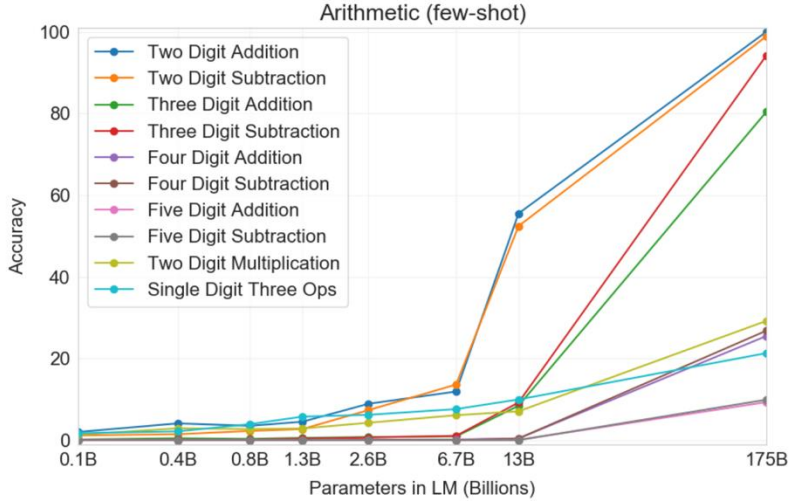




The Llama 3 Herd of Models (Meta, 2024)

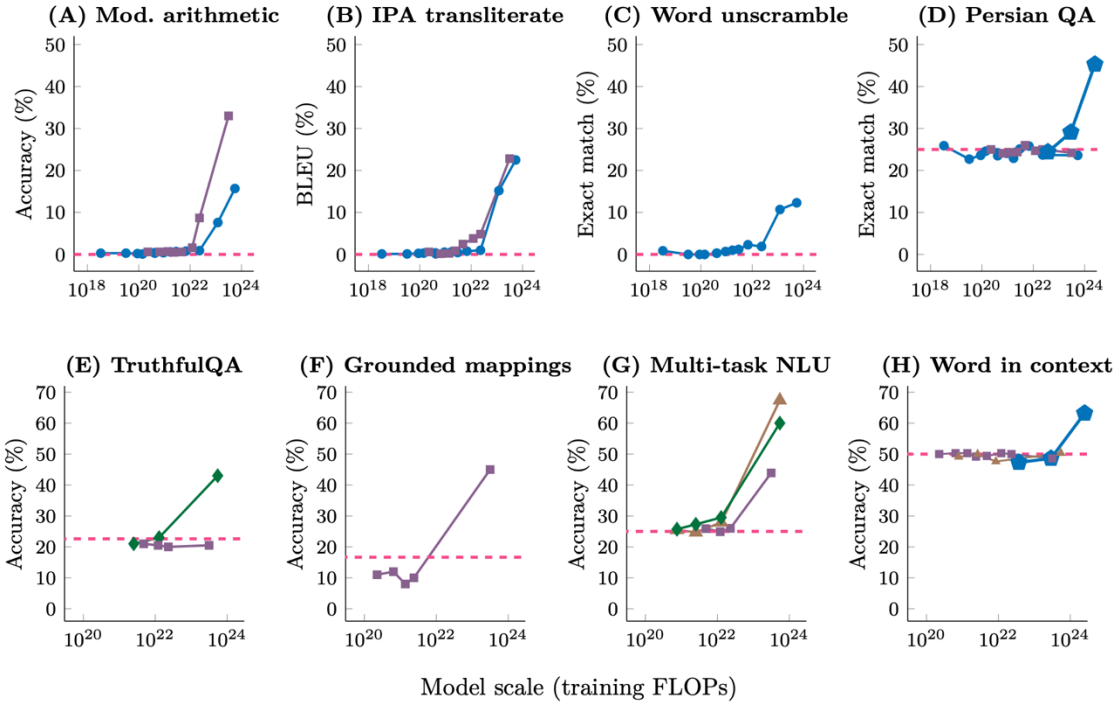
GPT-3 Hinted at Emergent Capabilities





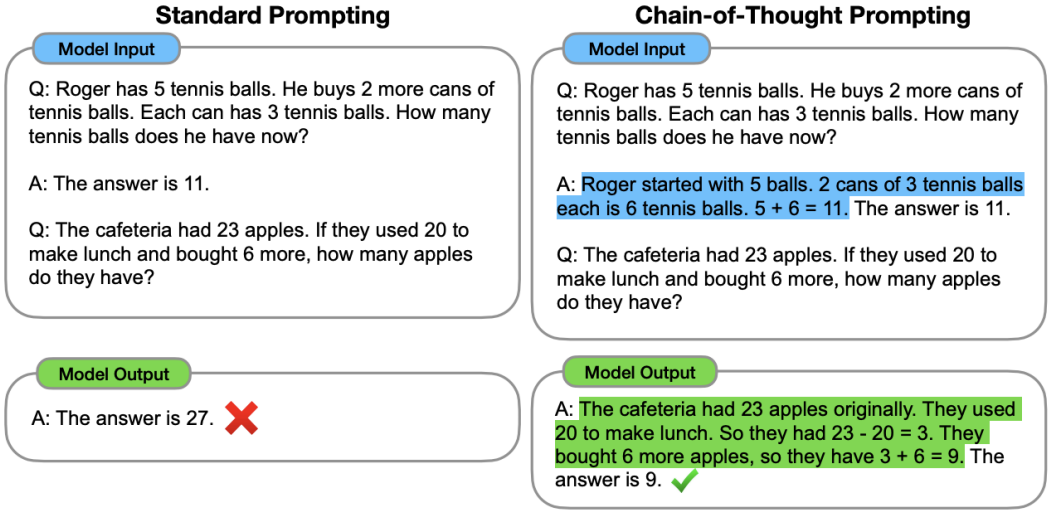
Language Models are Few-Shot Learners (OpenAI, 2020)

Emergence



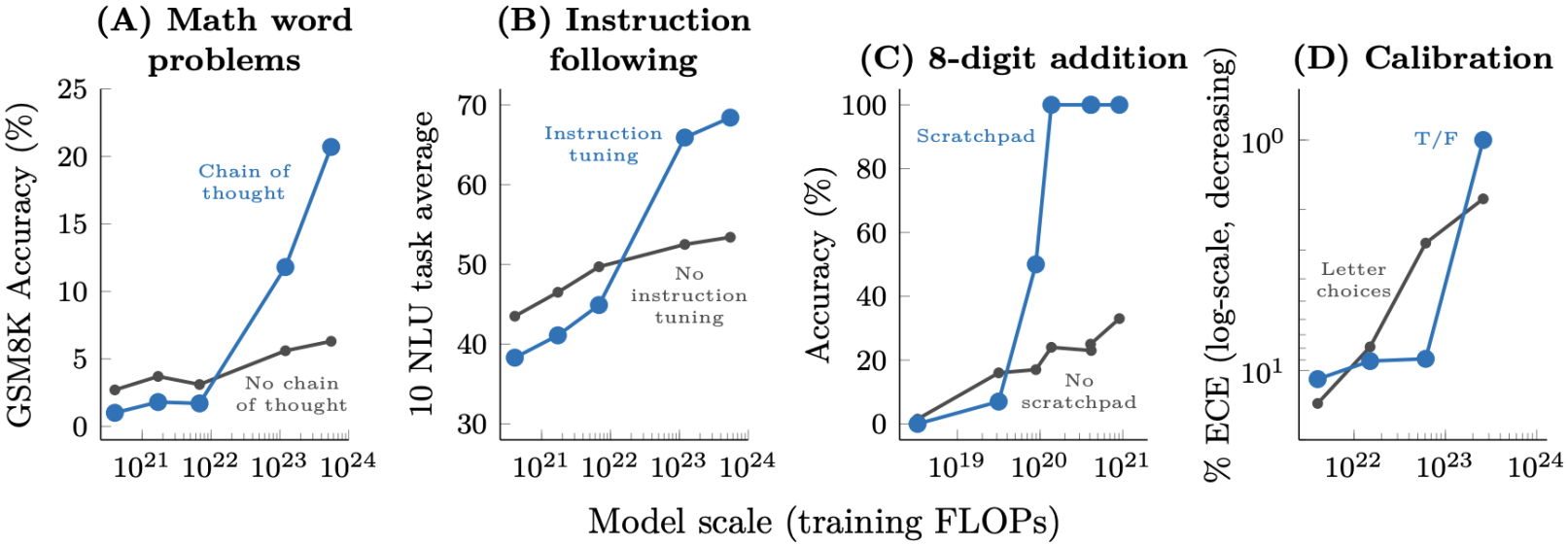
Emergent Capabilities of Large Language Models (Wei et al., 2022)

Emergence – Chain of Thought



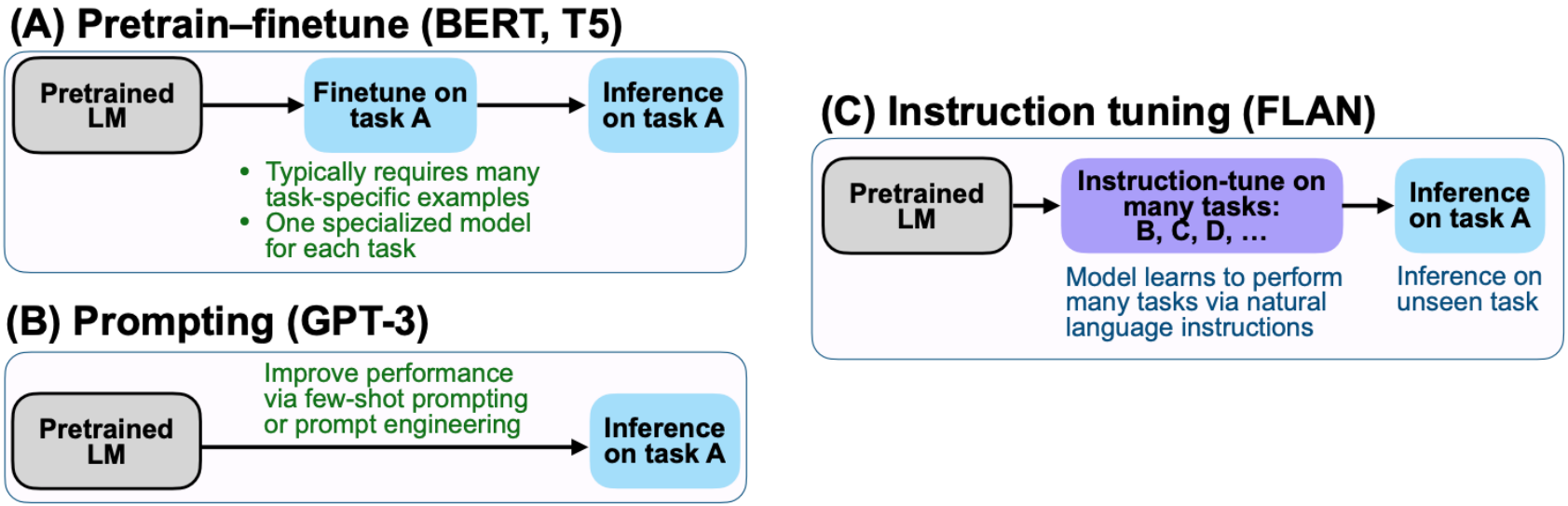
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (Wei et al., 2022)

Emergence – Chain of Thought



Emergent Capabilities of Large Language Models (Wei et al., 2022)

Instruction Tuning

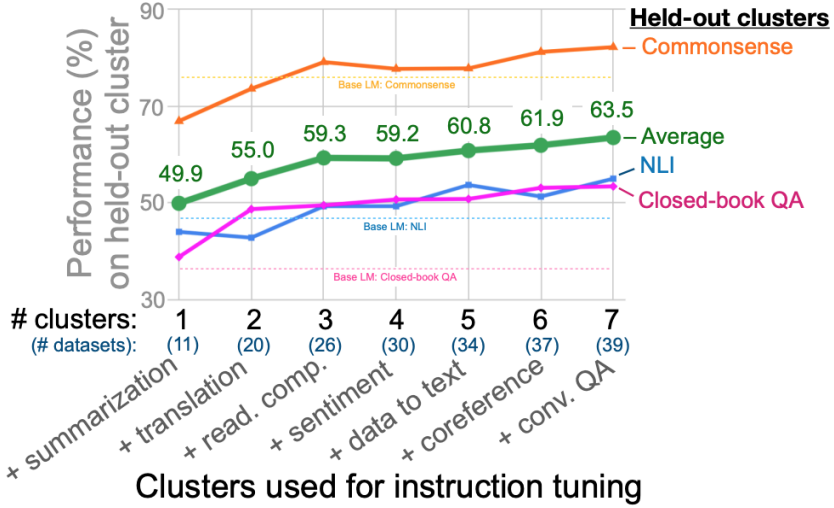


Finetuned Language Models are Zero-Shot Learners (Wei et al., 2022)

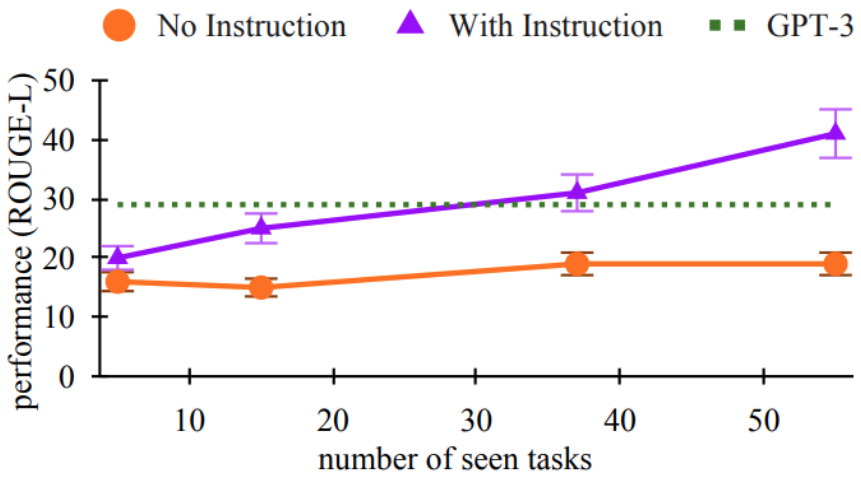
Instruction Tuning

Finetuned Language Models are Zero-Shot Learners (Wei et al., 2022)

Scaling Post-Training Data

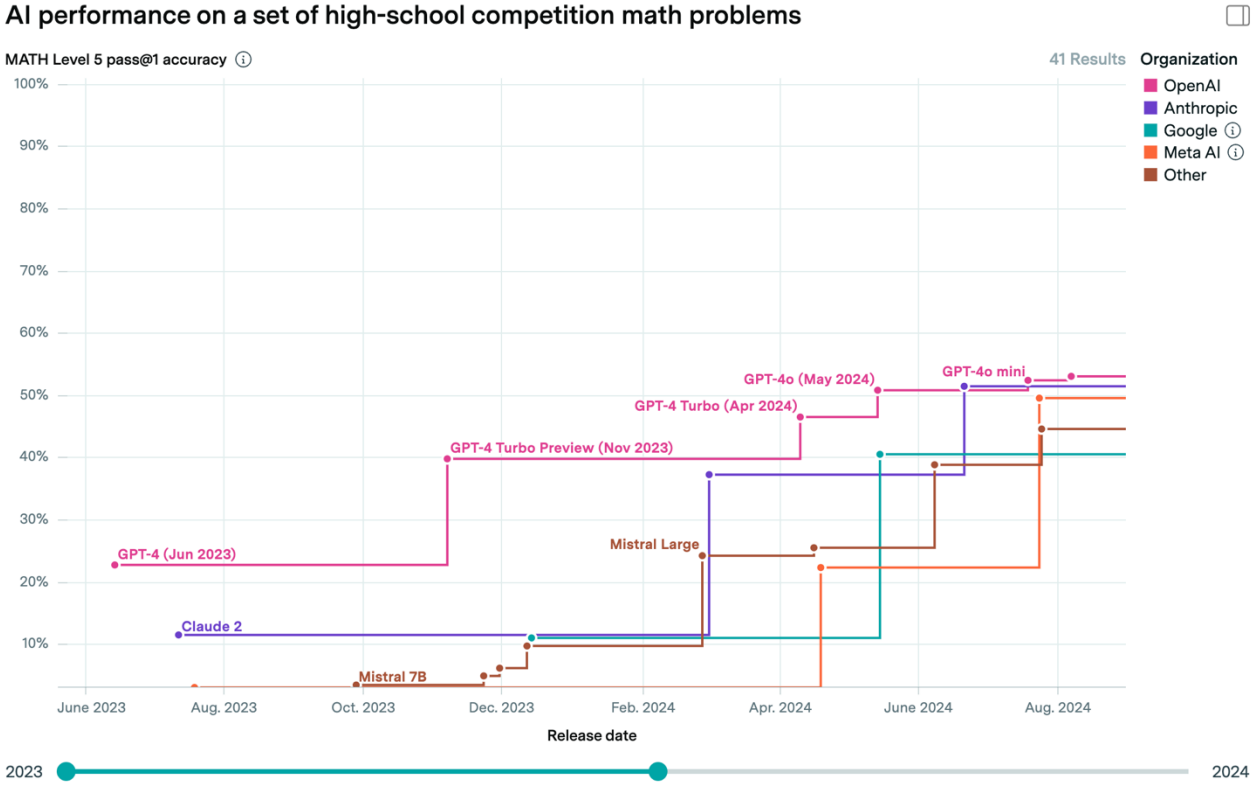


Finetuned Language Models are Zero-Shot Learners (Wei et al., 2022)

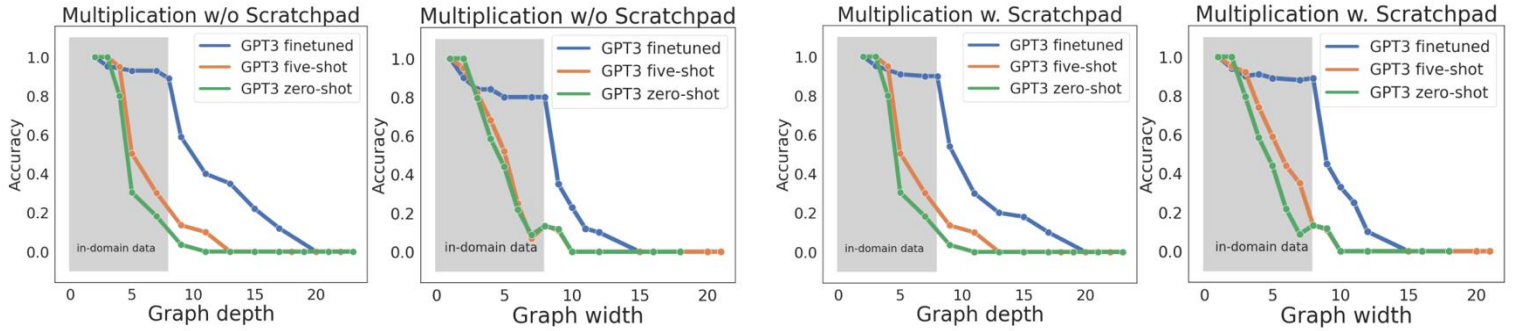


Cross-Task Generalization via Natural Language Crowdsourcing Instructions (Mishra et al., 2021)

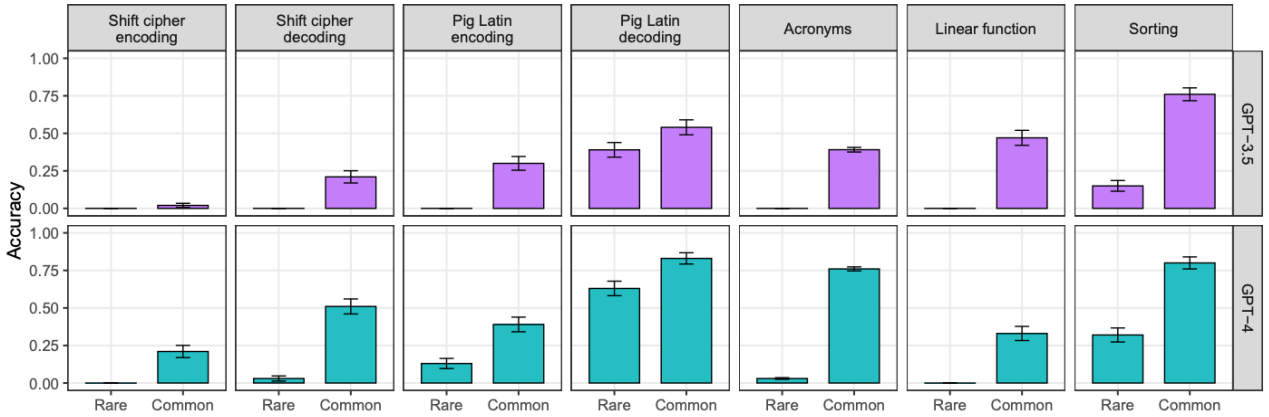
Advancement of LLM Capabilities (-mid 2024)



Training Data Bottlenecks Reasoning



Faith and Fate: Limits of Transformers on Compositionality (Dziri et al., 2023)



Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve (McCoy et al., 2023)

Training Data Bottlenecks Reasoning

Counting

Count the letters.

Input 1: iiiiiiiiiiiiiiiiiiiiiiiiiiiii
Correct: 30
✓ **GPT-4:** 30

Input 2: iiiiiiiiiiiiiiiiiiiiiiiiiiii
Correct: 29
✗ **GPT-4:** 30

Article swapping

Swap each article (*a*, *an*, or *the*) with the word before it.

Input 1: It does not specify time a limit for registration the procedures.
Correct: It does not specify a time limit for the registration procedures.
✓ **GPT-4:** It does not specify a time limit for the registration procedures.

Input 2: It few with it to lying take the get just a hands would kinds.
Correct: It few with it to lying the take get a just hands would kinds.
✗ **GPT-4:** It flew with a few kinds to take the lying just to get the hands.

Shift ciphers

Decode by shifting each letter 13 positions backward in the alphabet.

Input: Jryy, vg jnf abg rknpgyl cynaarq sebz gur ortvaavat.
Correct: Well, it was not exactly planned from the beginning.
✓ **GPT-4:** Well, it was not exactly planned from the beginning.

Decode by shifting each letter 12 positions backward in the alphabet.

Input: Iqxx, uf ime zaf qjmofxk bxmzzqp rday ftq nqsuzzuzs.
Correct: Well, it was not exactly planned from the beginning.
✗ **GPT-4:** Wait, we are not prepared for the apocalypse yet.

Linear functions

Multiply by 9/5 and add 32.

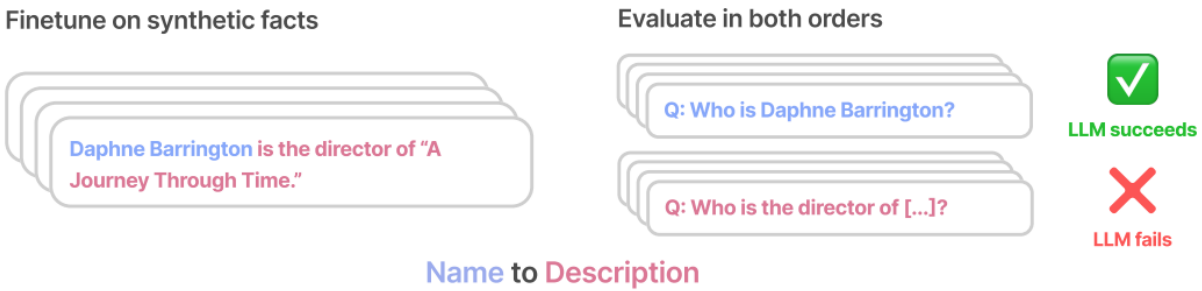
Input: 328
Correct: 622.4
✓ **GPT-4:** 622.4

Multiply by 7/5 and add 31.

Input: 328
Correct: 490.2
✗ **GPT-4:** 457.6

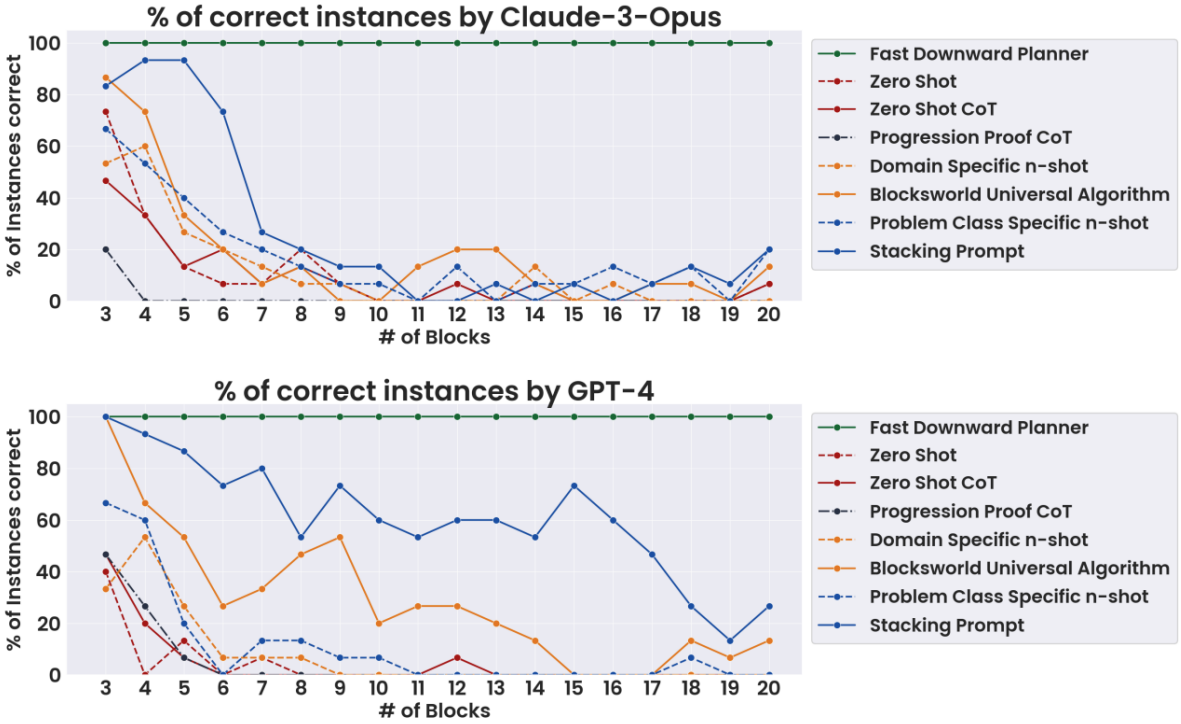
Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve (McCoy et al., 2023)

Metacognitive Skills (e.g., search) missing



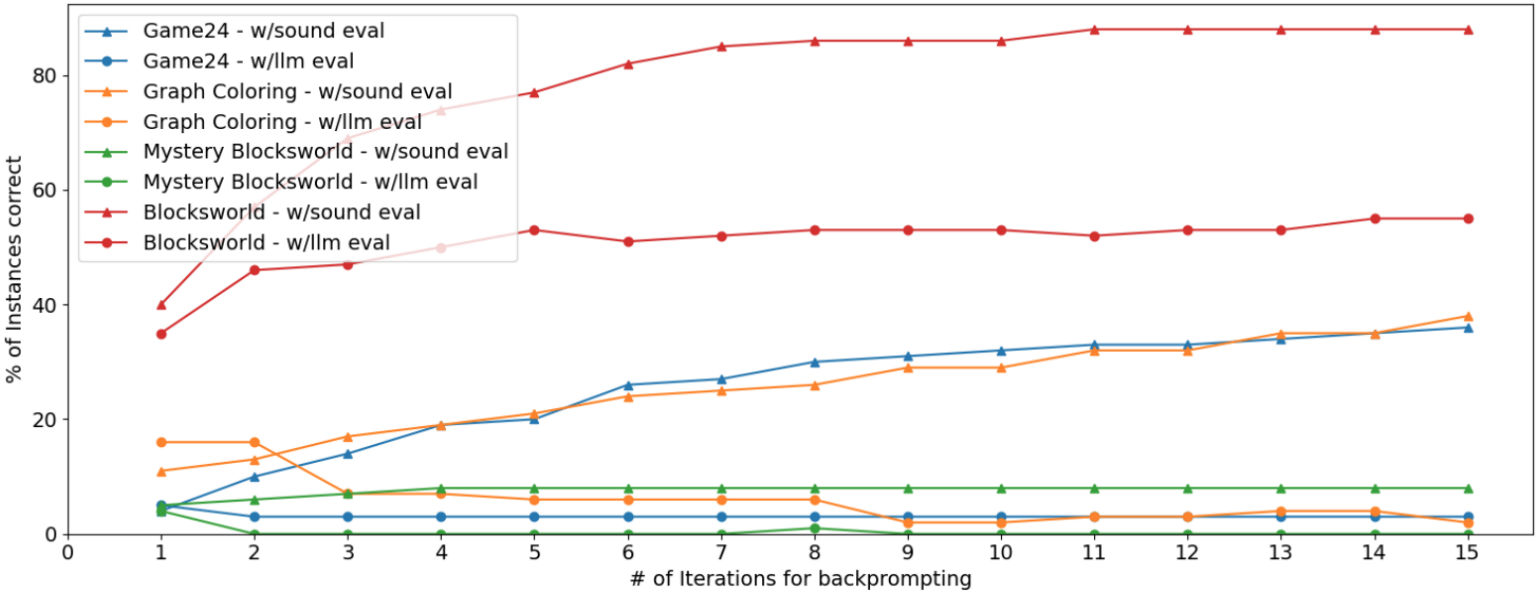
The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A" (Berglund et al., 2024)

Metacognitive Skills (e.g., search) missing



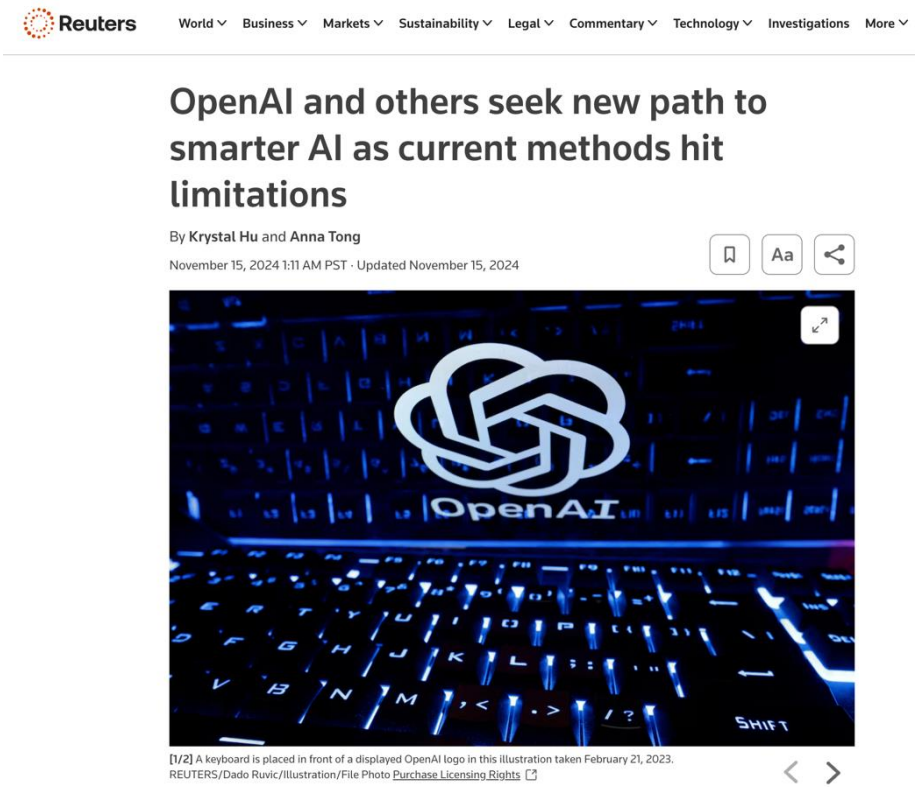
Chain of Thoughtlessness? An Analysis of CoT in Planning (Stechly et al., 2024)

Metacognitive Skills (e.g., search) missing



On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks (Stechly et al., 2024)

A New Scaling Axis was Needed



Reuters World Business Markets Sustainability Legal Commentary Technology Investigations More

OpenAI and others seek new path to smarter AI as current methods hit limitations

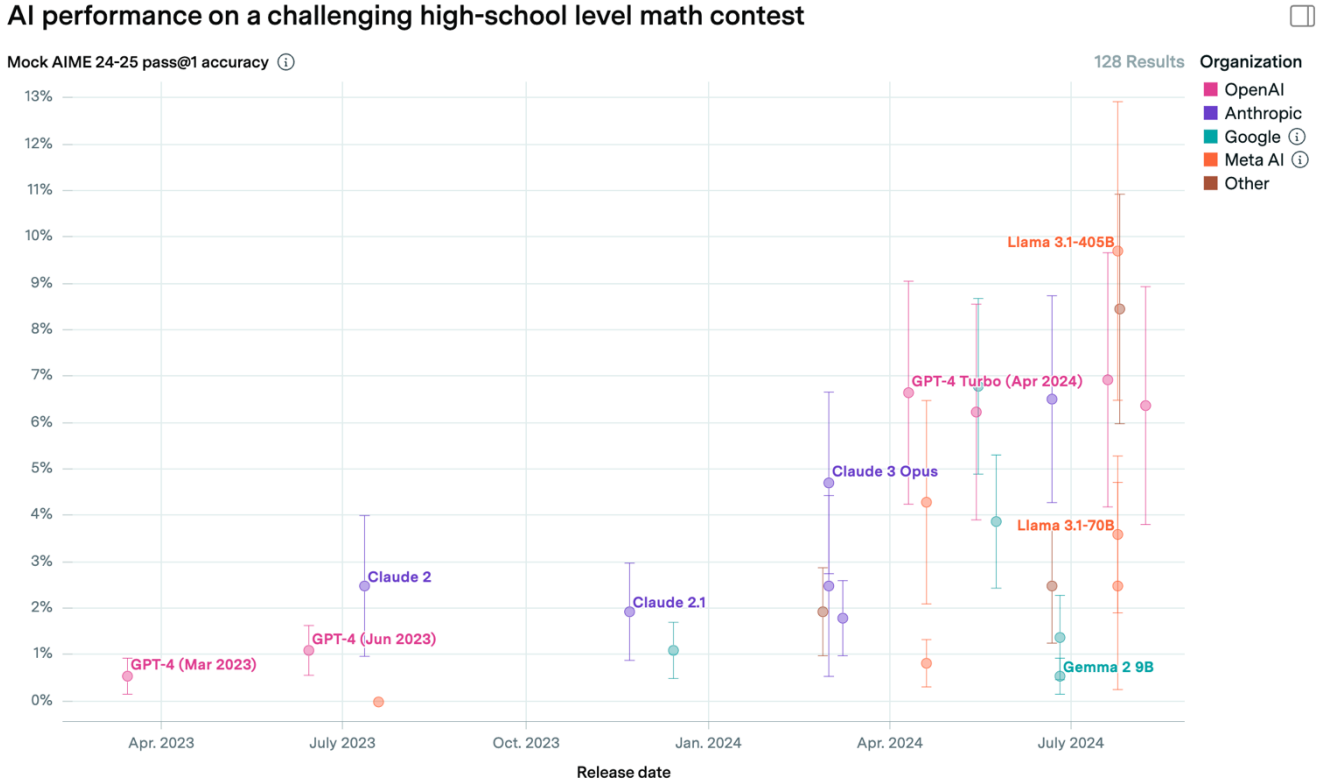
By Krystal Hu and Anna Tong

November 15, 2024 1:11 AM PST · Updated November 15, 2024

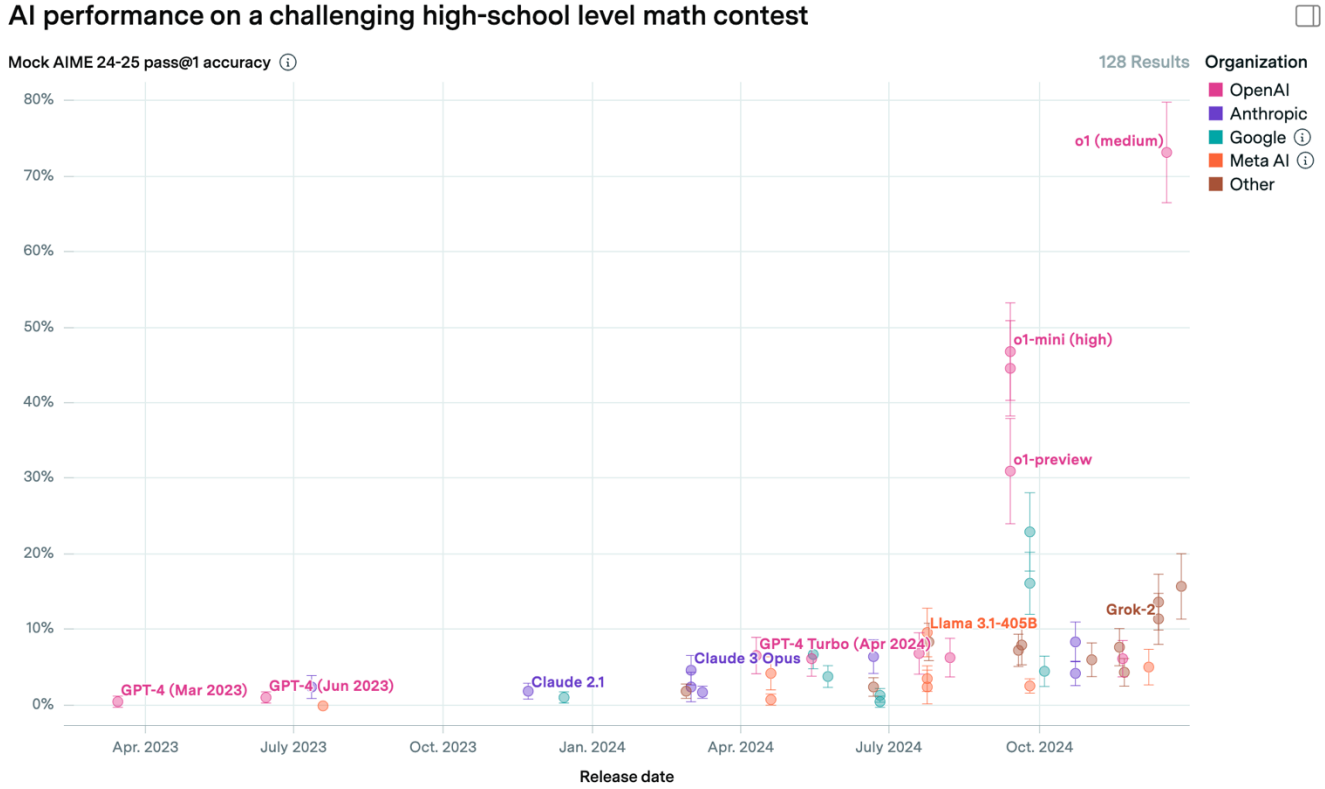
[1/2] A keyboard is placed in front of a displayed OpenAI logo in this illustration taken February 21, 2023.
REUTERS/Dado Ruvic/Illustration/File Photo Purchase Licensing Rights

https://www.reuters.com/technology/artificial-intelligence/openai-rivals-seek-new-path-smarter-ai-current-methods-hit-limitations-2024-11-11/

Things moved from here…



To here.. How?


AI performance on a challenging high-school level math contest
Mock AIME 24-25 pass@1 accuracy
128 Results
Organization
OpenAI
Anthropic
Google
Meta AI
Other
80%
70%
60%
50%
40%
30%
20%
10%
0%
o1 (medium)
o1-mini (high)
o1-preview
Grok-2
Llama 3.1-405B
GPT-4 Turbo (Apr 2024)
Claude 3 Opus
Claude 2.1
GPT-4 (Jun 2023)
GPT-4 (Mar 2023)
Apr. 2023
July 2023
Oct. 2023
Jan. 2024
Apr. 2024
July 2024
Oct. 2024
Release date

Reinforcement Learning – an overview

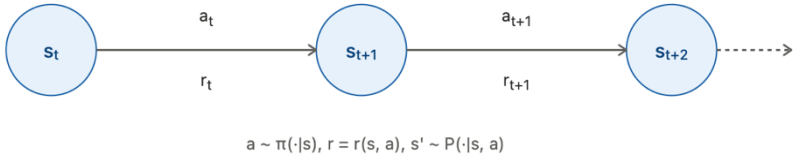
Markov Decision Process (MDP)
- State s_t
- Action a_t
- Transition Dynamics $P(s'|s,a)$
- Reward Function $r(s,a)$
- Discount factor γ

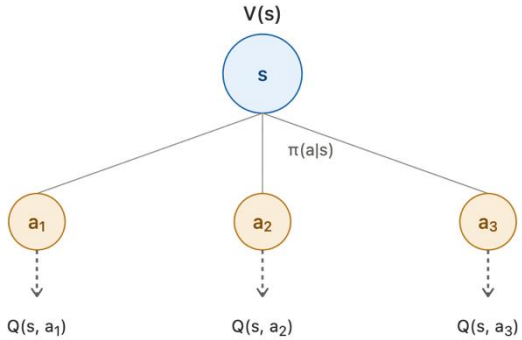
Policy $\pi(a|s)$: maps states to distributions over actions

At each timestep, record (s_t, a_t, r_t)

Gain $G_t = \Sigma_{k\geq 0}\gamma^k r_{t+k}$

Value function $V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$





Reinforcement Learning – an overview

Q-function $Q^\pi(s,a)$ measures the expected return starting from s after committing to action a

Definition: $Q^\pi(s,a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$

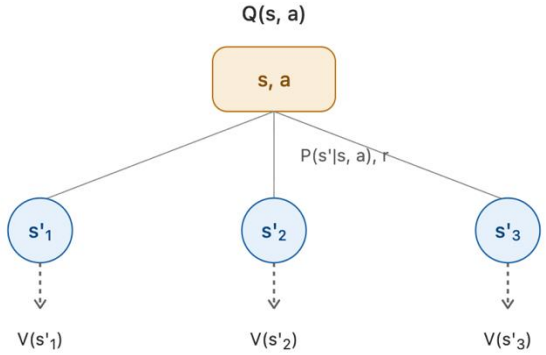
Relationship to value function:

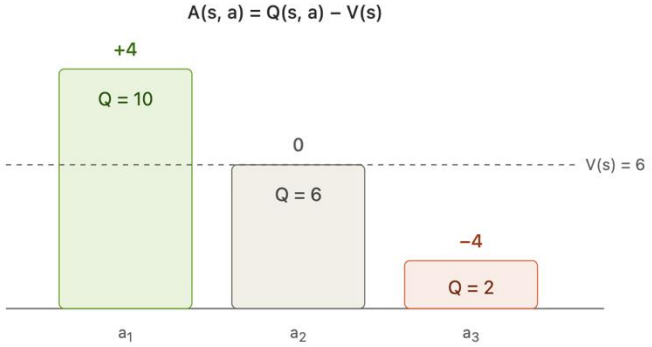
$$V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a)]$$

Advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

Adaptation to policy gradients (REINFORCE):

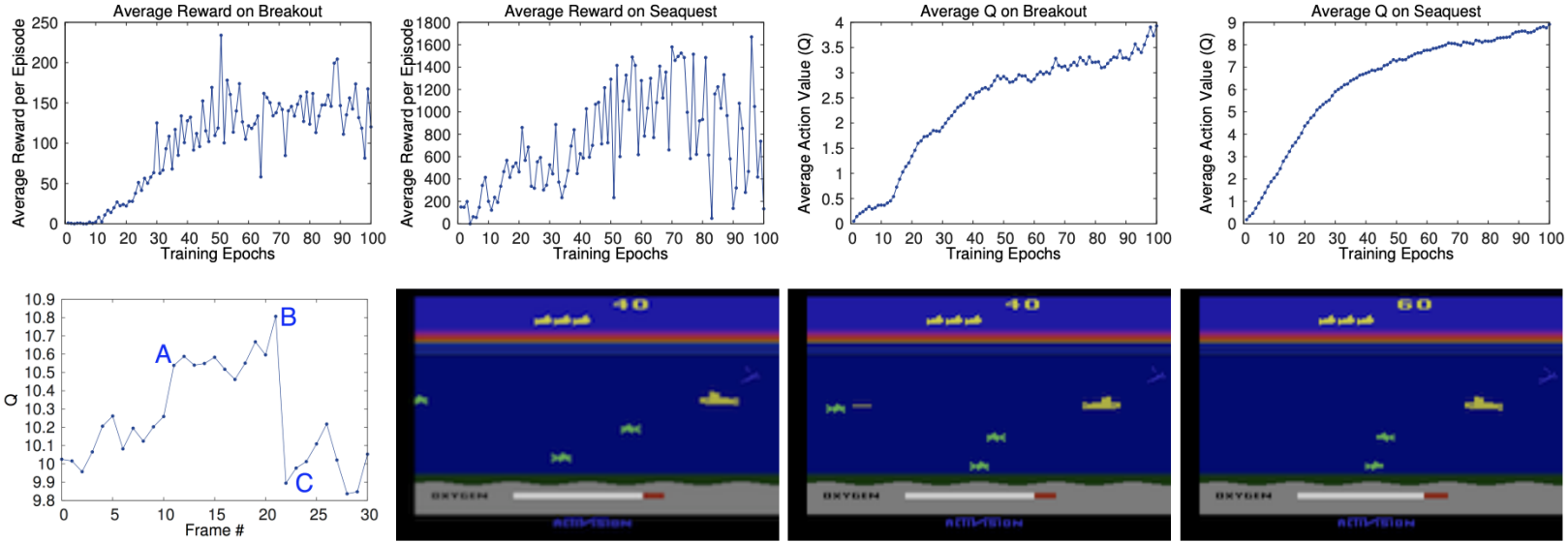
$$\hat{g} = \mathbb{E}[\nabla \log \pi(a|s)\, A^\pi(s,a)]$$





Deep Q-Networks – Introduction of DL to RL

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right]$$



Playing Atari with Deep Reinforcement Learning (Mnih et al., 2013 (DeepMind))

Parametrization of the Q-function

Algorithm 1 Deep Q-learning with Experience Replay

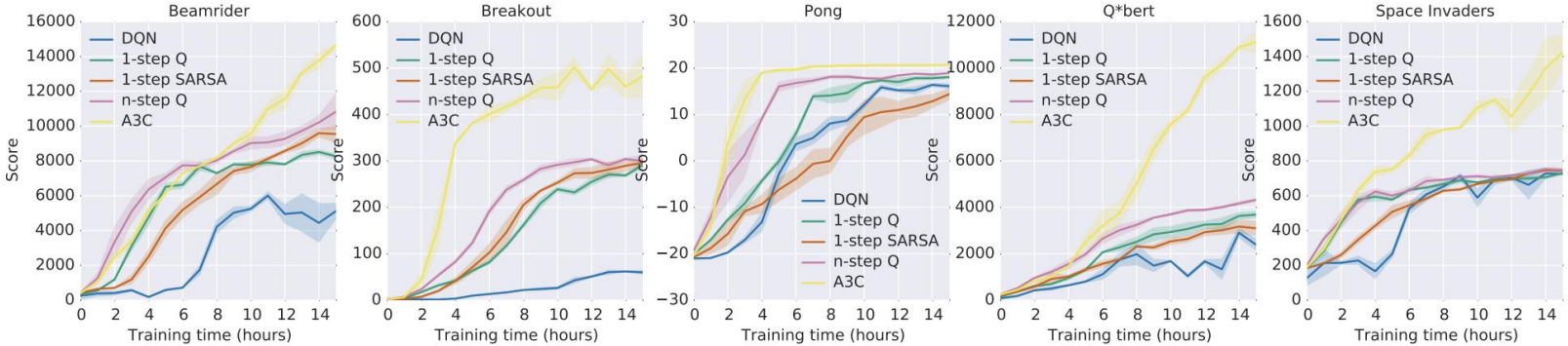
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Parametrization of Actor-Critic Networks

repeat
  Perform a_t according to policy $\pi(a_t|s_t;\theta')$
  Receive reward r_t and new state s_{t+1}
  $t \leftarrow t+1$
  $T \leftarrow T+1$
until terminal s_t **or** $t - t_{start} == t_{max}$

$$R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$$

for $i \in \{t-1, \ldots, t_{start}\}$ **do**
  $R \leftarrow r_i + \gamma R$
  Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
  Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
end for
Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

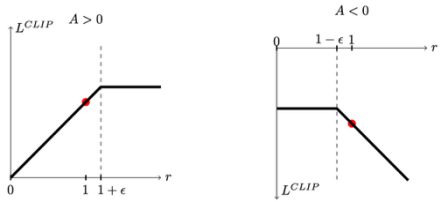


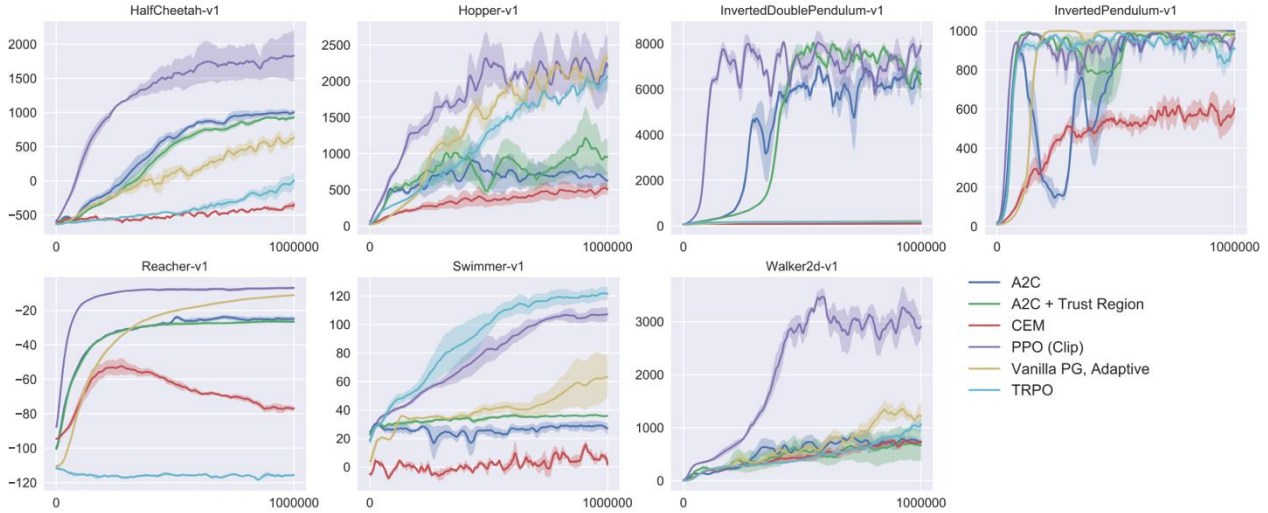
Asynchronous Methods for Deep Reinforcement Learning (Mnih et al., 2016 (DeepMind))

Proximal Policy Optimization

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$





Proximal Policy Optimization Algorithms *(Schulman et al., 2017 (OpenAI))*

AlphaGo: Parametrization of Policy and Value

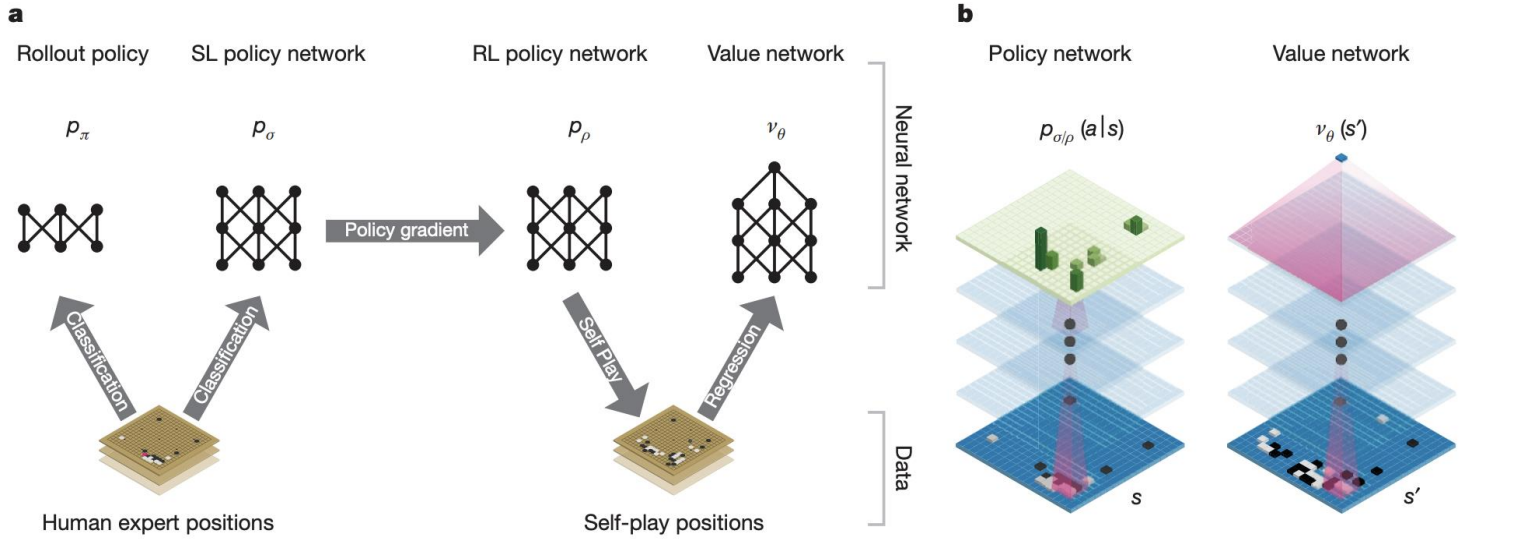


Figure 1 | Neural network training pipeline and architecture. a, A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network p_ρ is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network v_θ is trained by regression to predict the expected outcome (that is, whether the current player wins) in positions from the self-play data set. **b**, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position s as its input, passes it through many convolutional layers with parameters σ (SL policy network) or ρ (RL policy network), and outputs a probability distribution $p_\sigma(a|s)$ or $p_\rho(a|s)$ over legal moves a, represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters θ, but outputs a scalar value $v_\theta(s')$ that predicts the expected outcome in position s'.

AlphaGo: Search Enables Better Outcomes

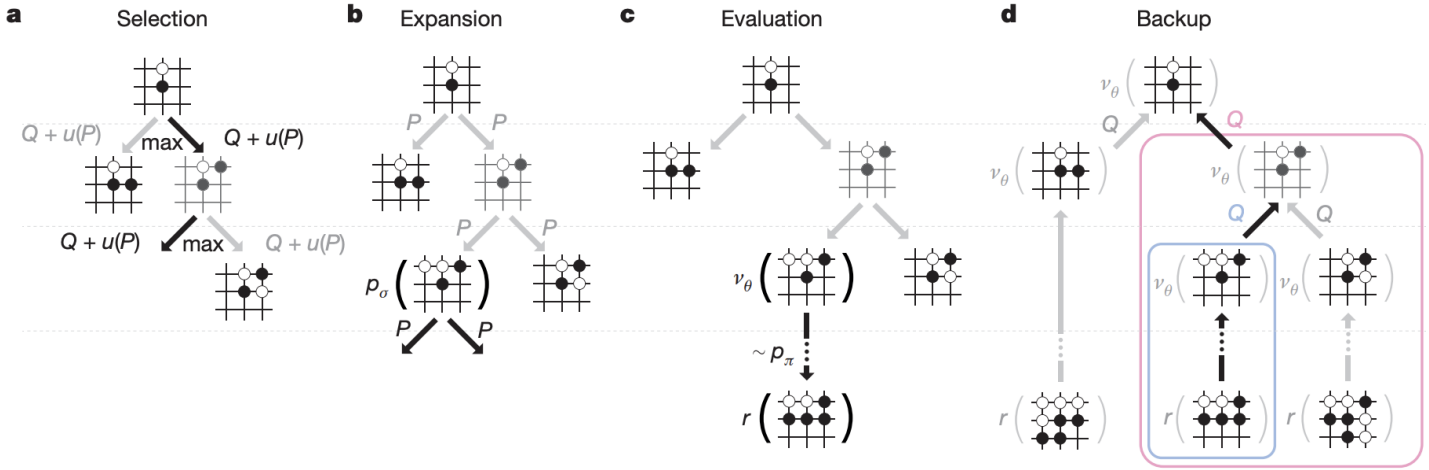
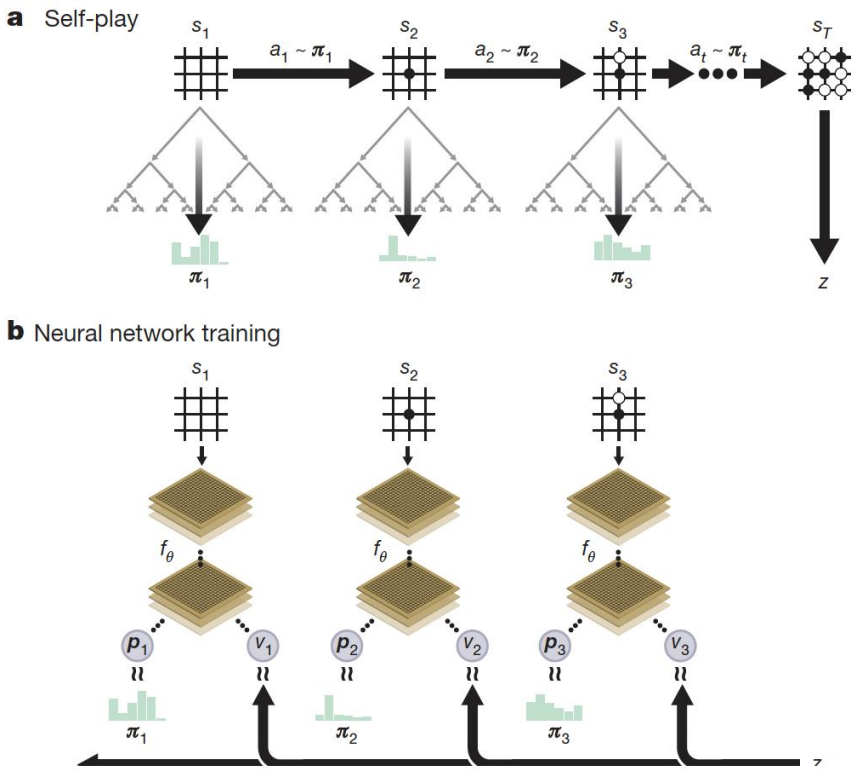
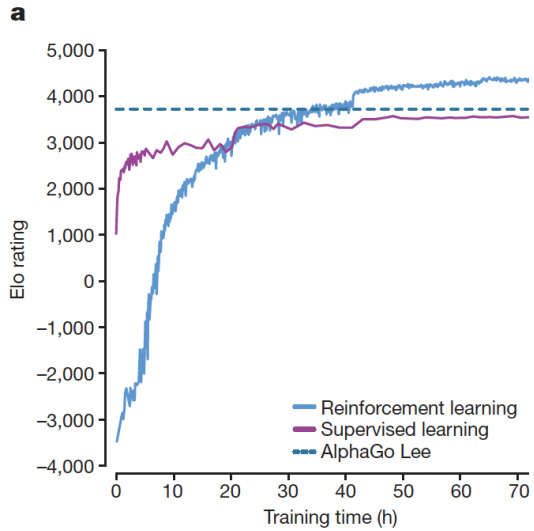


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

AlphaGo Zero: 0 → 1 (Search + Self-Play + RL)



(1) Single network predicts action probability and state value
(2) No explicit rollouts conducted – estimation at leaf node
(3) MCTS as the policy improvement operator



$$(\boldsymbol{p}, v) = f_\theta(s) \text{ and } l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$

Mastering the game of Go without human knowledge (DeepMind, 2017)

All Signs Pointed to Search and Learning

The Bitter Lesson

Rich Sutton

March 13, 2019

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.

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, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. 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.

In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked---they tried to put that knowledge in their systems---but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use.

In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.

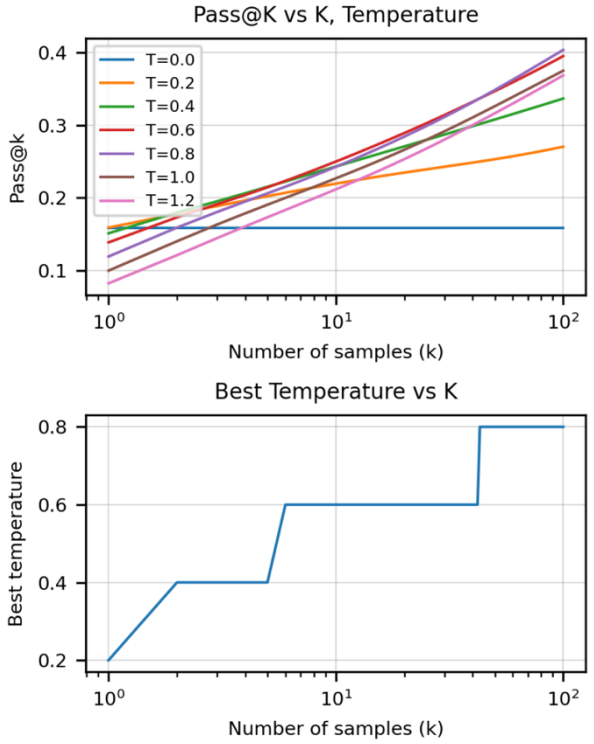
This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does 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.

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*.

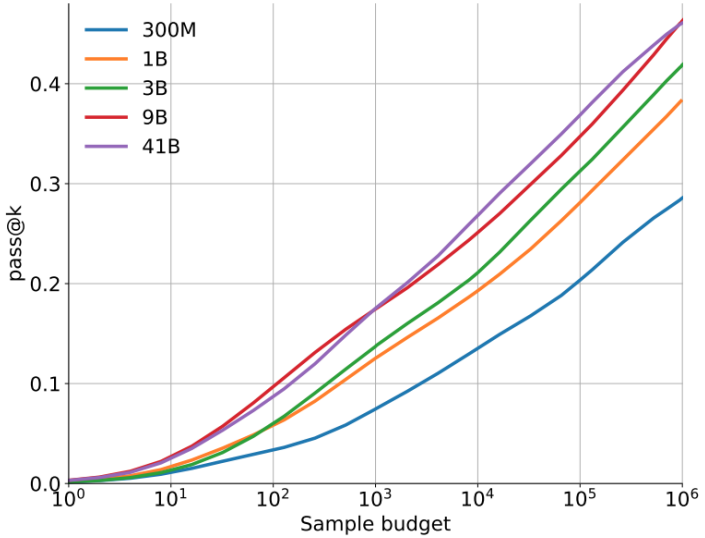
The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; 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.

Concluded that search and learning are the only two methods that seemed to scale arbitrarily with more computation!

What is the search space for LMs?

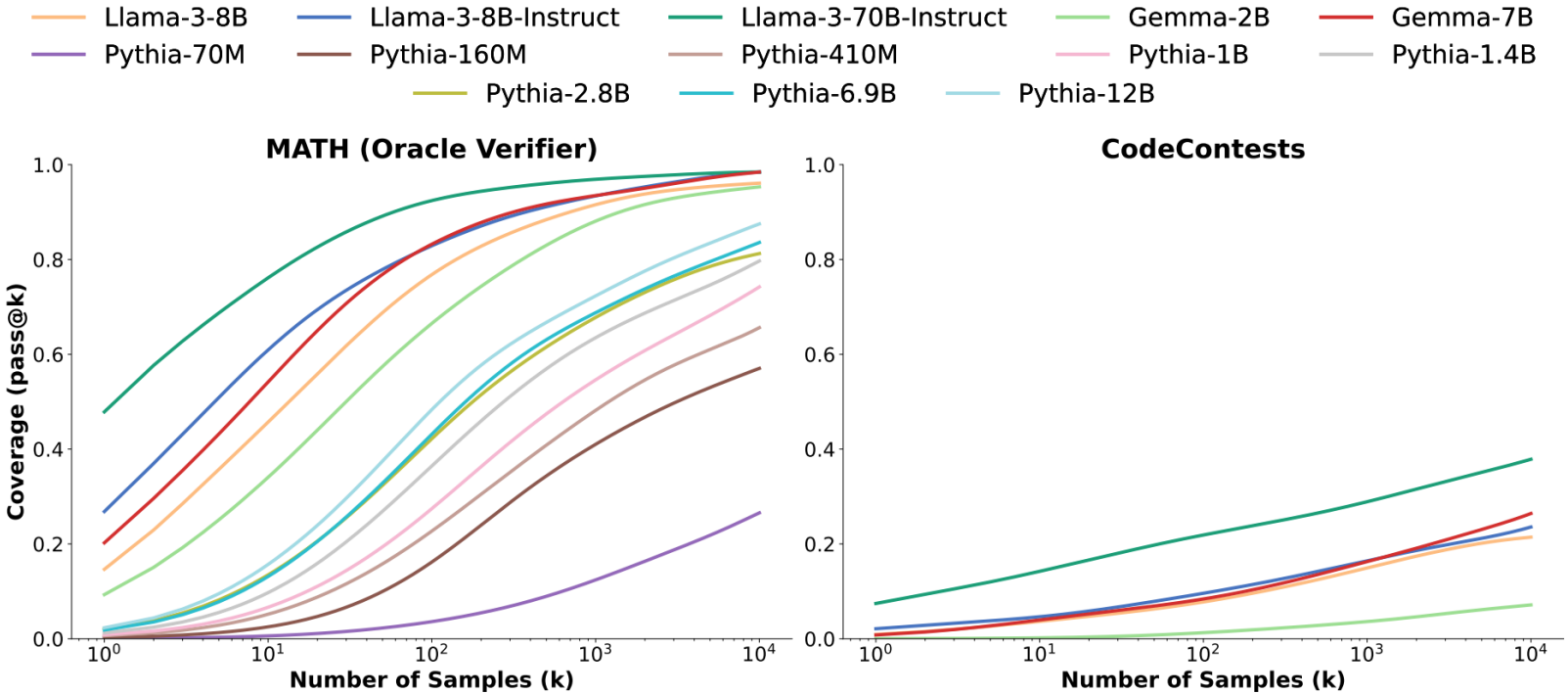


Evaluating Large Language Models Trained on Code (OpenAI, 2021)



Competition-Level Code Generation with AlphaCode (DeepMind, 2022)

What is the search space for LMs?

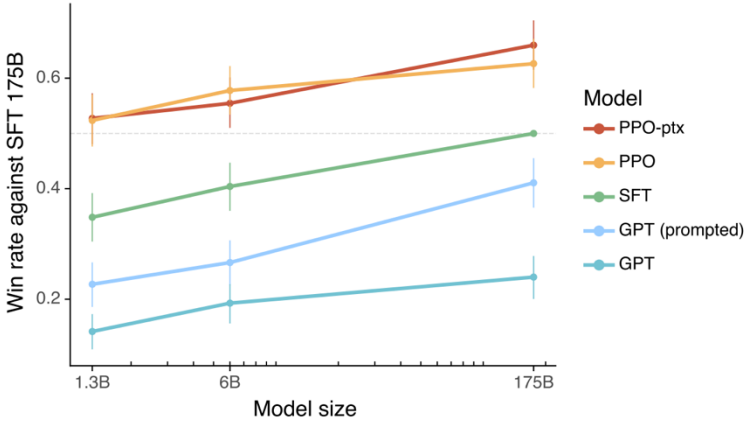


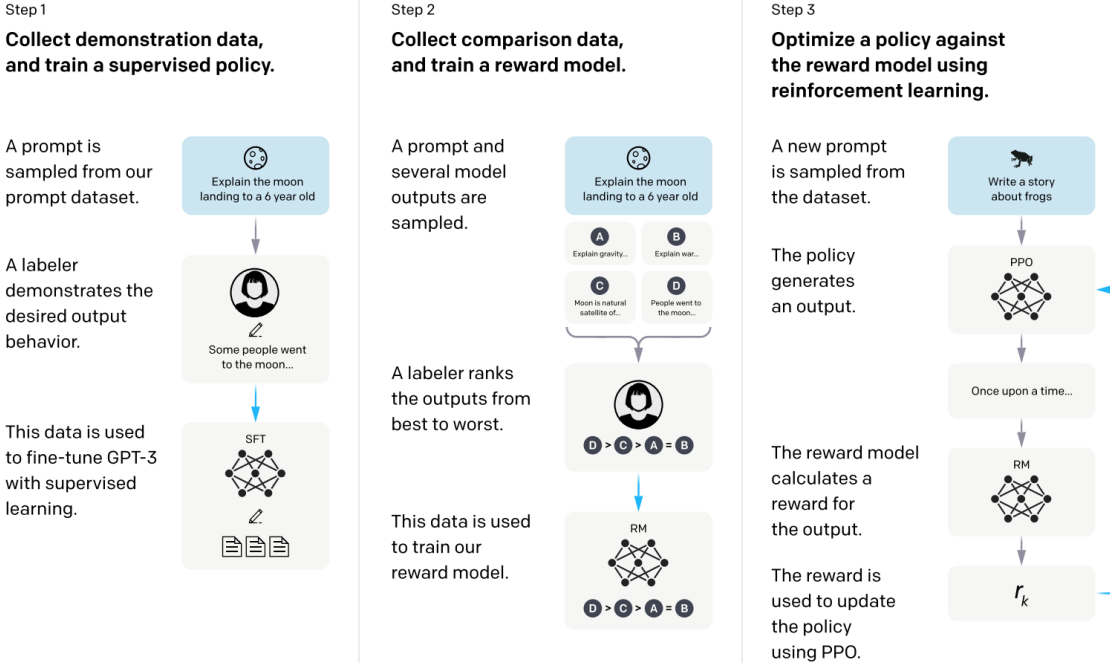
Large Language Monkeys: Scaling Inference Compute with Repeated Sampling (Brown et al., 2024)

Extending to Language Models…

AlphaZero	LLM (phase 1)	LLM (phase 2)
Policy	Pre-trained LLM, token distribution	Pre-trained LLM, multiple reasoning step candidates
Value	Value Model, Verifier	PRM, Value Model, Verifier
MCTS	Autoregressive generation	Autoregressive generation, optionally with environment feedback
Self-Play generates (state, improved-policy, outcome) targets	Rollouts generate (prompt, trajectory, reward) targets	Rollouts generate (prompt, trajectory, reward) targets
Policy distillation from MCTS targets	PPO update on reward-weighted trajectories	PPO/GRPO update on reward-weighted trajectories
Simulator (game rules)	Token concatenation	Depends (whether external environment exists)

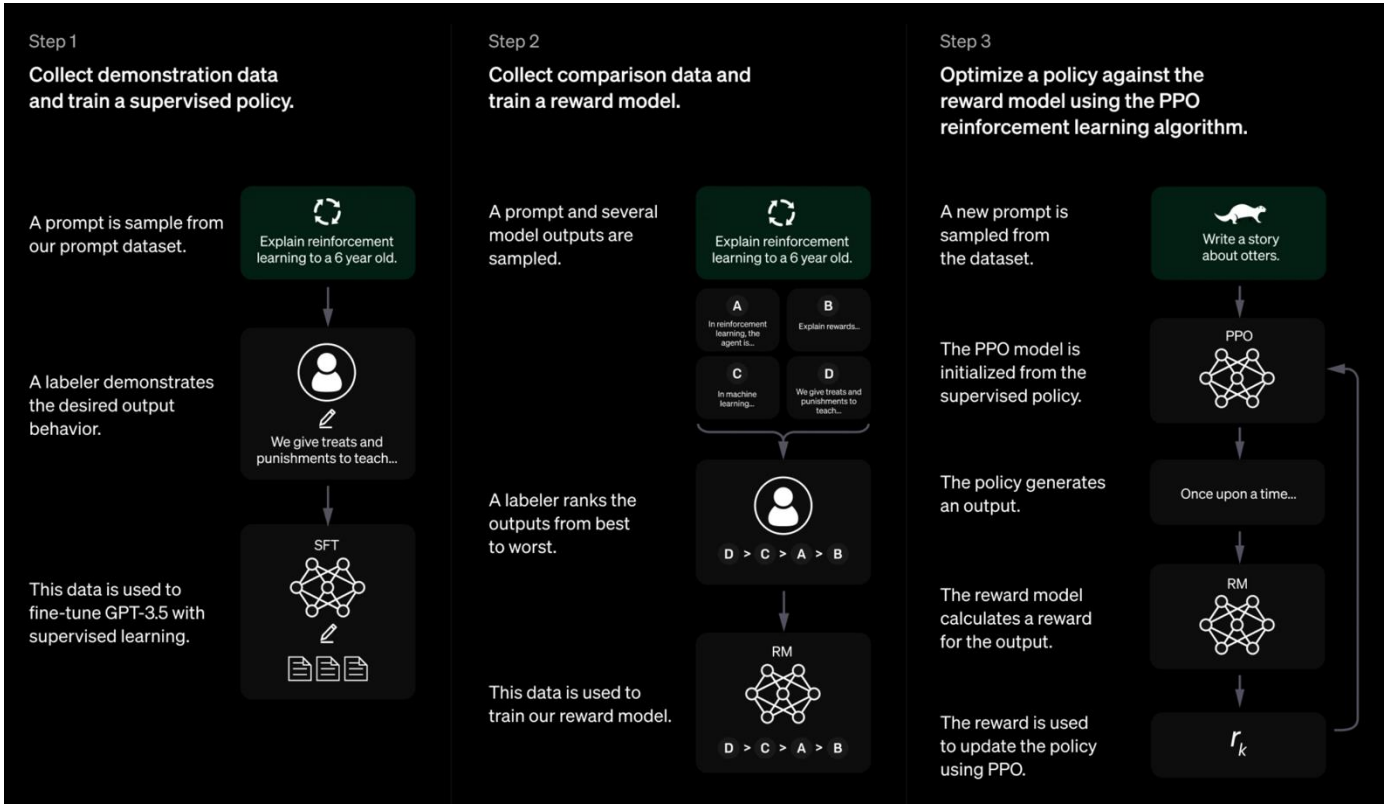
InstructGPT – Reinforcement Learning



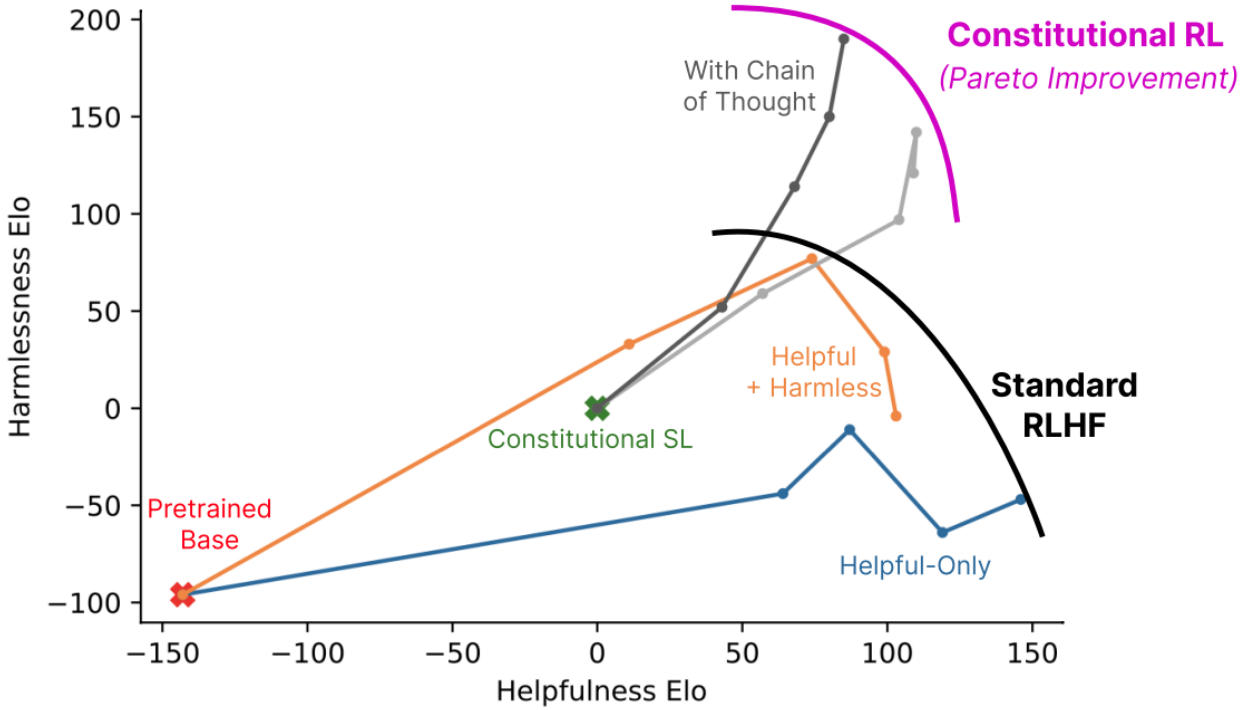


Training language models to follow instructions with human feedback (OpenAI, 2022)

ChatGPT 3.5 release (Nov 2022)

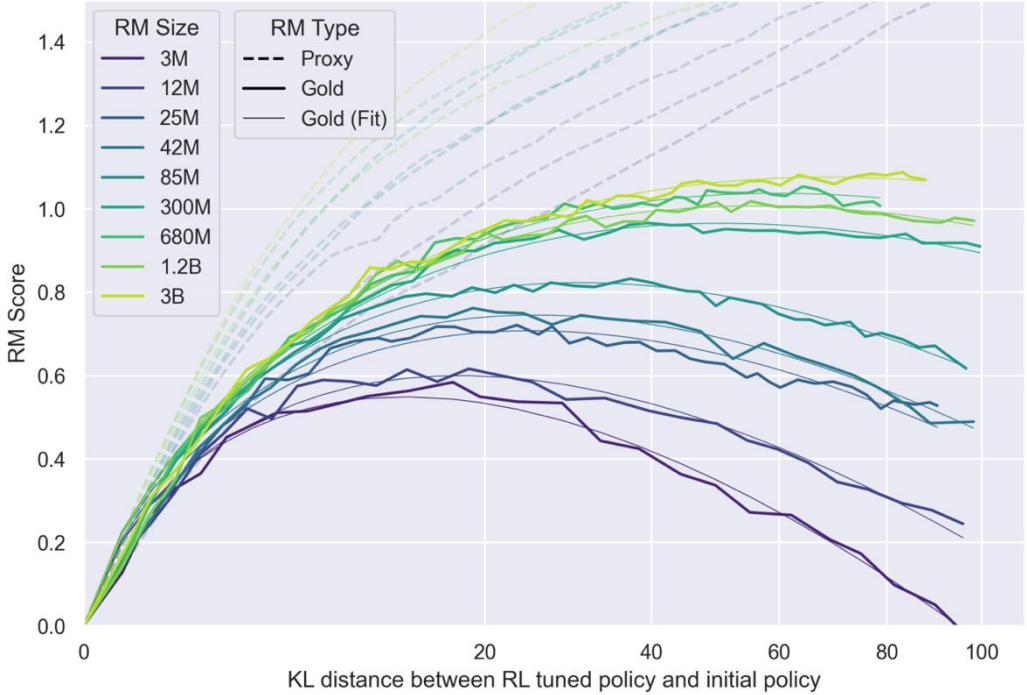


Rewards from Model-Generated Judgments



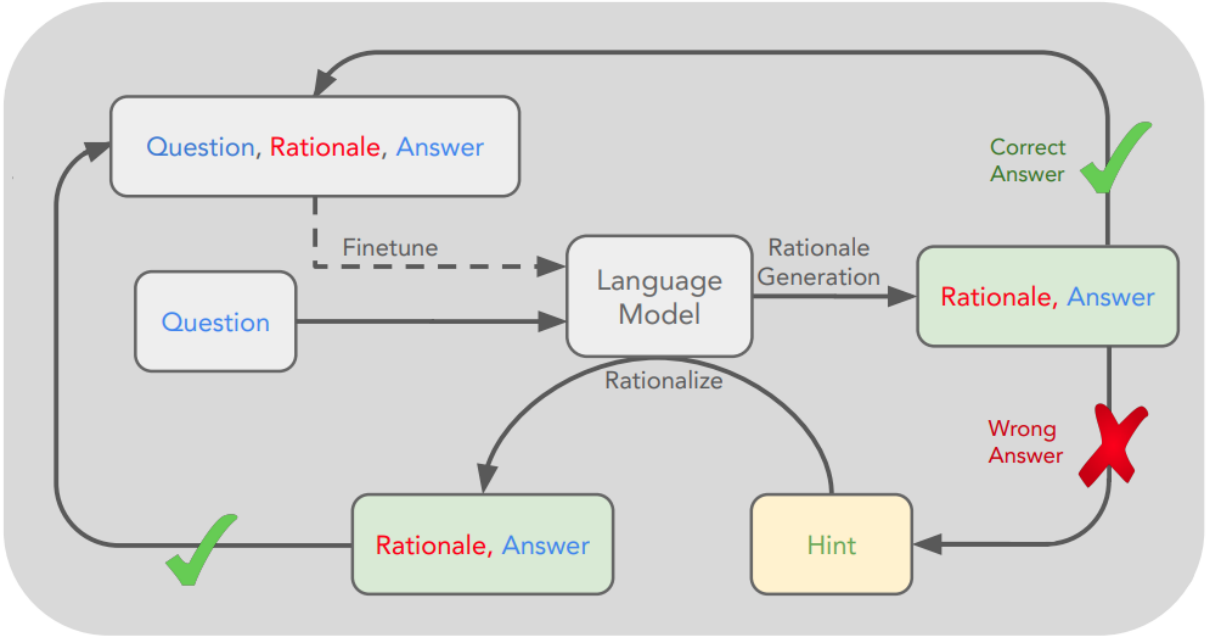
Constitutional AI: Harmlessness from AI Feedback (Anthropic, 2022)

Do RMs provide reliable reward signals?



Scaling Laws for Reward Model Overoptimization (OpenAI, 2022)

Using Ground-Truth for Reasoning



Q: What can be used to carry a small dog?
Answer Choices:
(a) swimming pool
(b) basket
(c) dog show
(d) backyard
(e) own home
A: The answer must be something that can be used to carry a small dog. Baskets are designed to hold things. Therefore, the answer is basket (b).

STaR (Self-Taught Reasoner): Bootstrapping Reasoning With Reasoning (Zelikman et al., 2022)

Using Ground-Truth for Reasoning

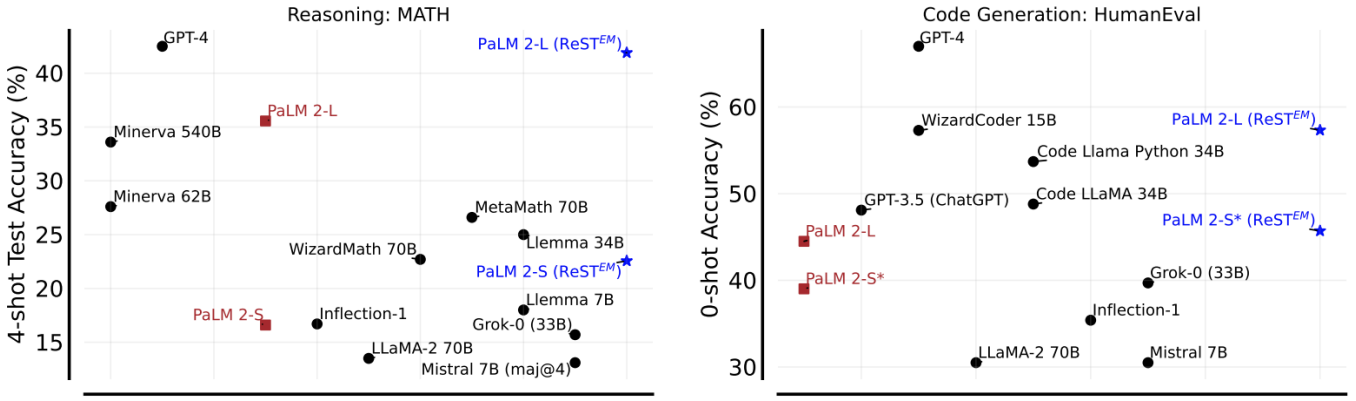


Figure 1 | Self-training with ReST^{EM} substantially improves test performance of PaLM 2 models on two challenging benchmarks: MATH and HumanEval. Results for other models are shown for general progress on these tasks and are typically not comparable due to difference in model scales. GPT-4 results are taken from Bubeck et al. (2023). The x-axis approximately denotes release time (not to scale).

Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models (DeepMind, 2023)

Using Ground-Truth for Reasoning

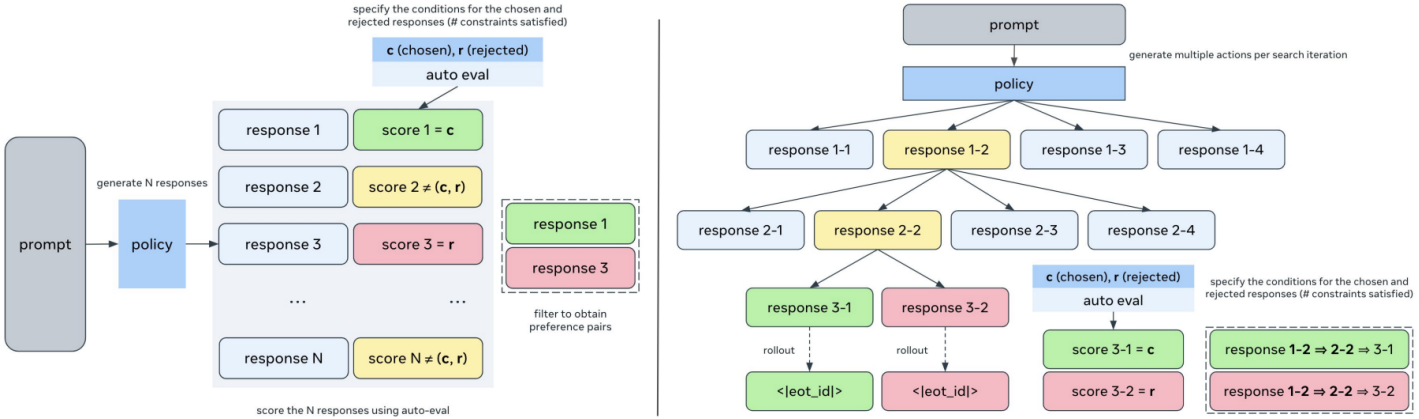
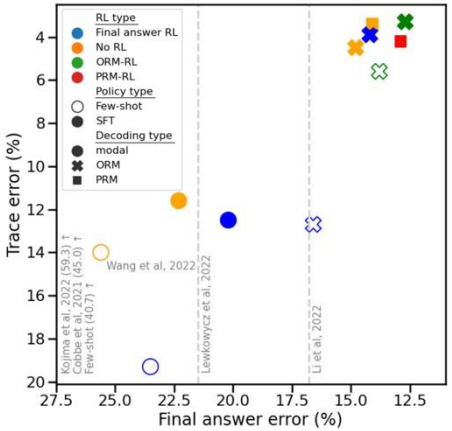


Figure 1: Automatically curating preference pairs via rejection sampling (RS, left) and Monte Carlo Tree Search (MCTS, right). **RS**: We independently sample N different outputs from the policy, score each output with a verifier and take (high, low) scoring responses as the (chosen, rejected) pairs. **MCTS**: We perform tree search with the policy while generating multiple actions. Then, we use the rollouts from sibling nodes with (high, low) reward scores as the (chosen, rejected) pairs to obtain preference pairs with common prefixes up to the parent nodes.

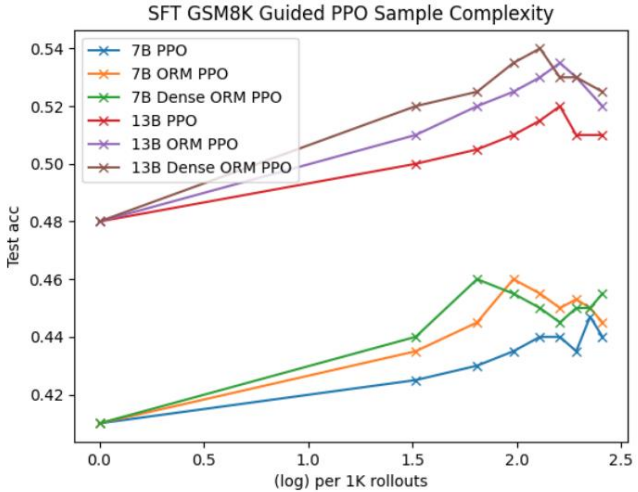
A Systematic Examination of Preference Learning through the Lens of Instruction-Following (Kim et al., 2024)

A Misguided Understanding of Ground-Truth



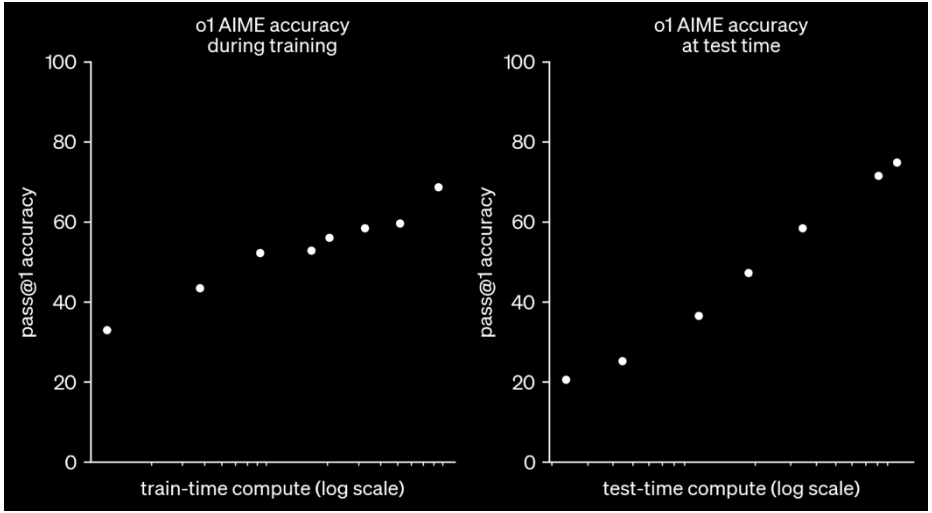
Model	Error rate Greedy	Majority	RM-weighted
Few-shot	54.3	41.4	27.8
Few-shot + Final-answer RL	36.4	23.8	16.6
Few-shot + ORM-RL	31.5	18.8	13.8
SFT	41.1	22.3	14.1
SFT + Final-answer RL	35.0	20.2	14.2
SFT + ORM-RL	31.2	17.8	12.7
SFT + PRM-RL	34.4	17.6	12.9

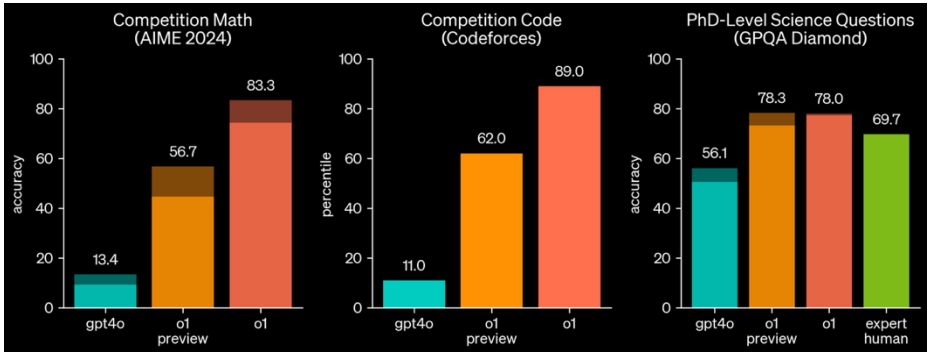
Solving math word problems with process and outcome-based feedback (DeepMind, 2022)



Teaching Large Language Models to Reason with Reinforcement Learning (Meta, 2024)

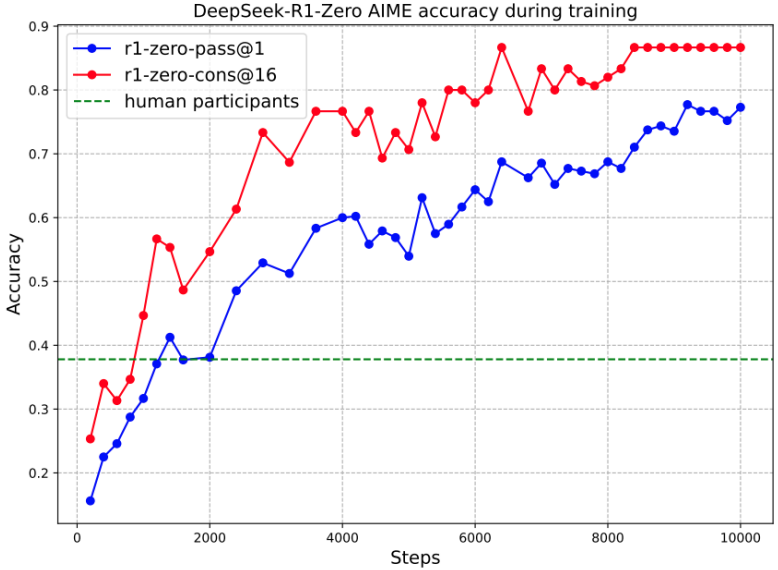
RL for Reasoning – OpenAI o1

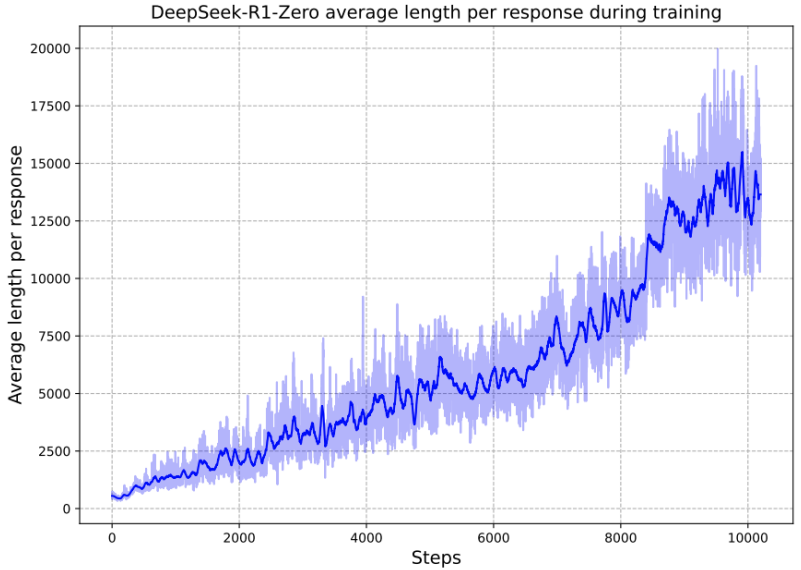




https://openai.com/index/learning-to-reason-with-llms/

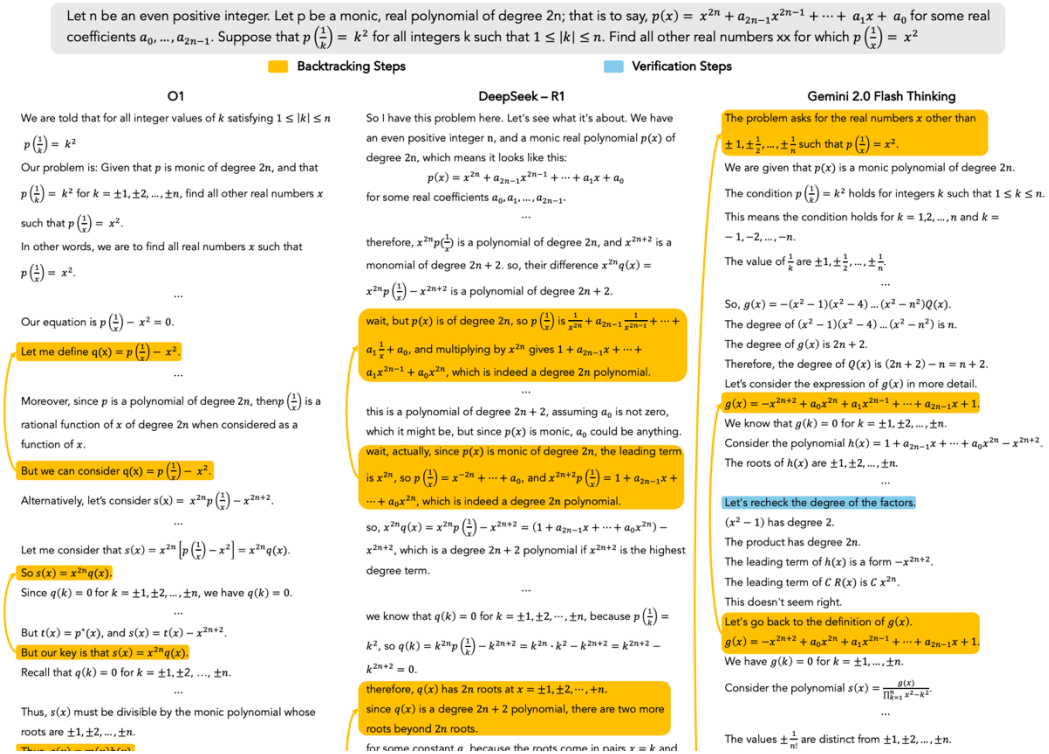
RL for Reasoning – DeepSeek R1





DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning (DeepSeek, 2025)

Search Behavior in Thinking Traces

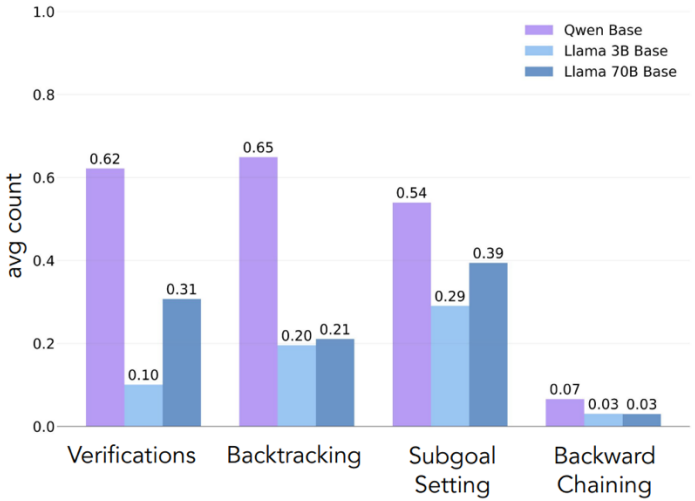


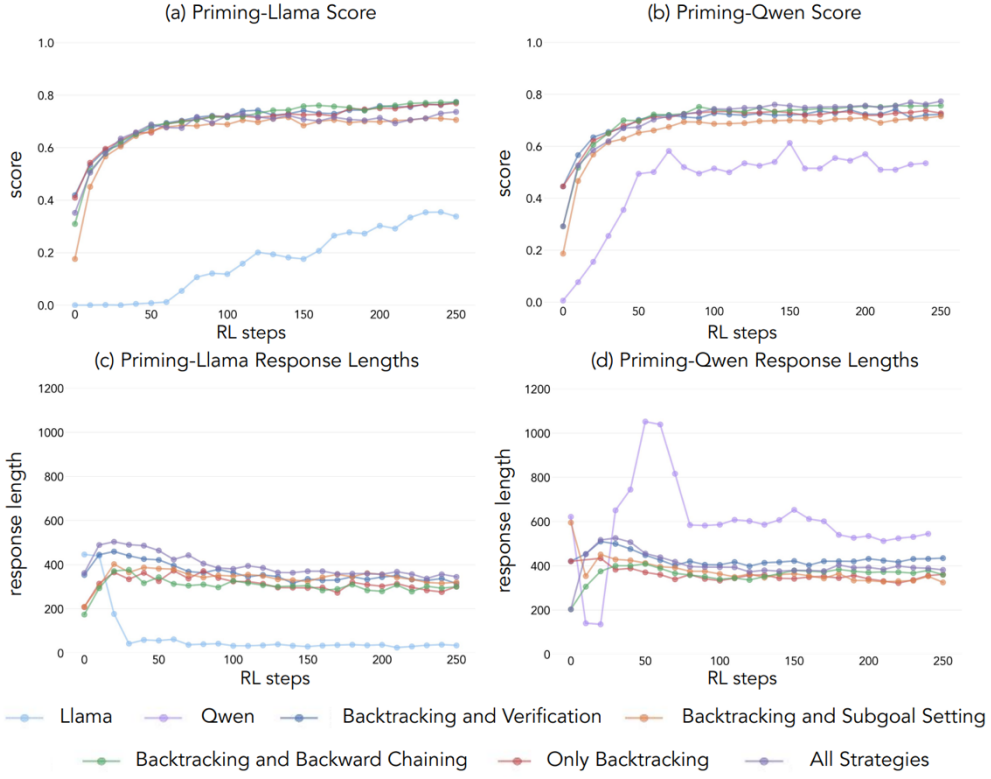
Towards System 2 Reasoning in LLMs: Learning How to Think With Meta Chain-of-Thought (Xiang et al., 2025)

What is Needed to Unlock Thinking? (1)

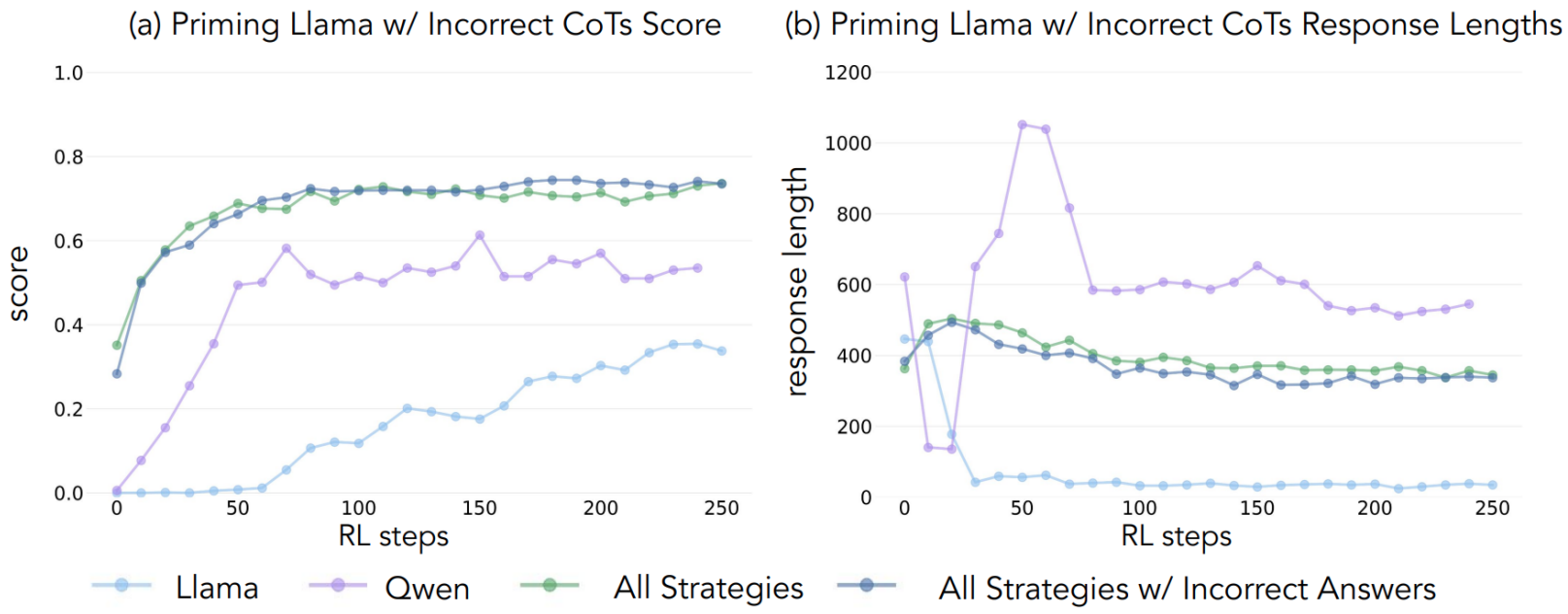
Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs (Gandhi et al., 2025)

What is Needed to Unlock Thinking? (1)

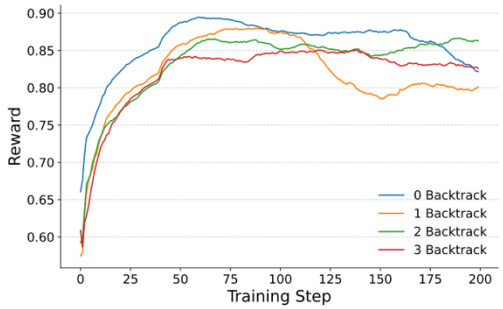




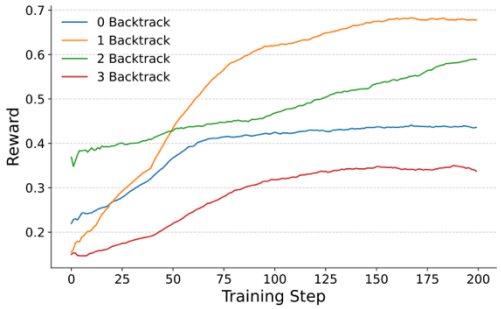
What is Needed to Unlock Thinking? (1)



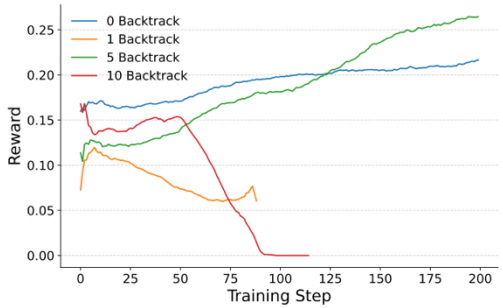
What is Needed to Unlock Thinking? (2)



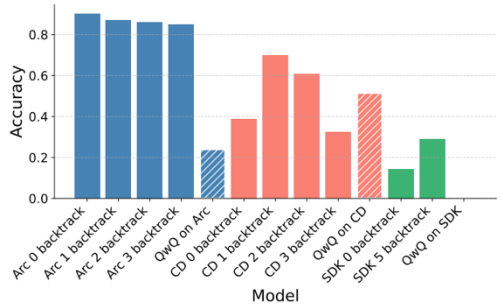
(a) Arc 1D



(b) Countdown



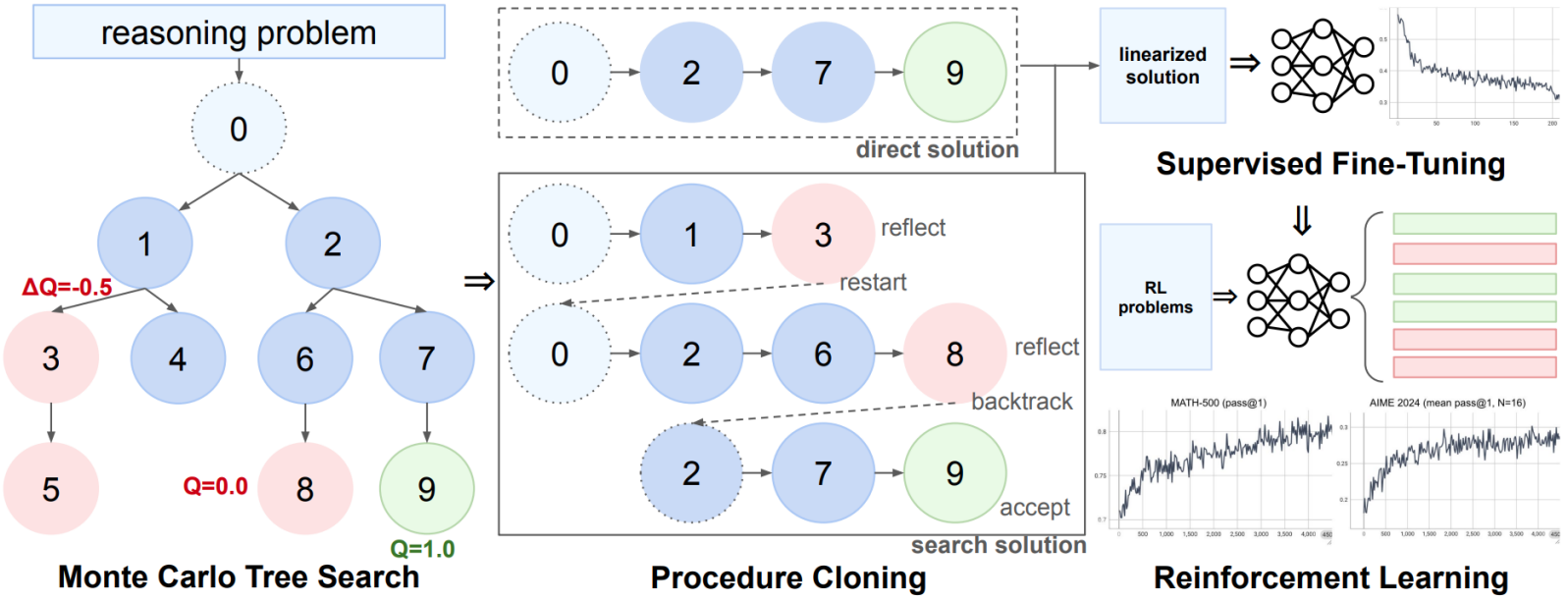
(c) Sudoku



(d) In domain evaluation

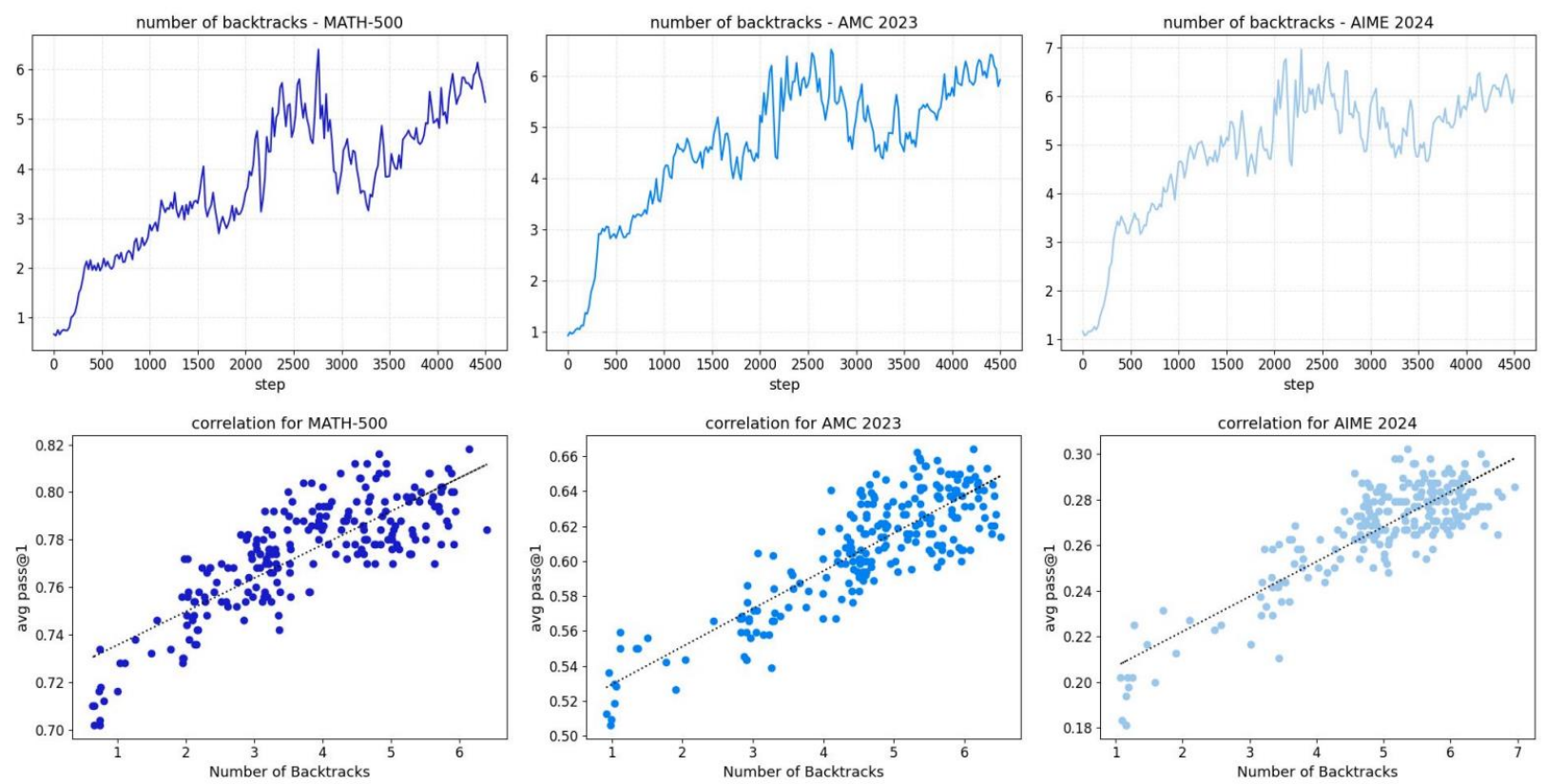
How Much Backtracking is Enough? Exploring the Interplay of SFT and RL in Enhancing LLM Reasoning (Cai et al., 2025)

What is Needed to Unlock Thinking? (3)



ASTRO: Teaching Language Models to Reason by Reflecting and Backtracking In-Context (Kim et al., 2025)

What is Needed to Unlock Thinking? (3)



What is Needed to Unlock Thinking? (3)

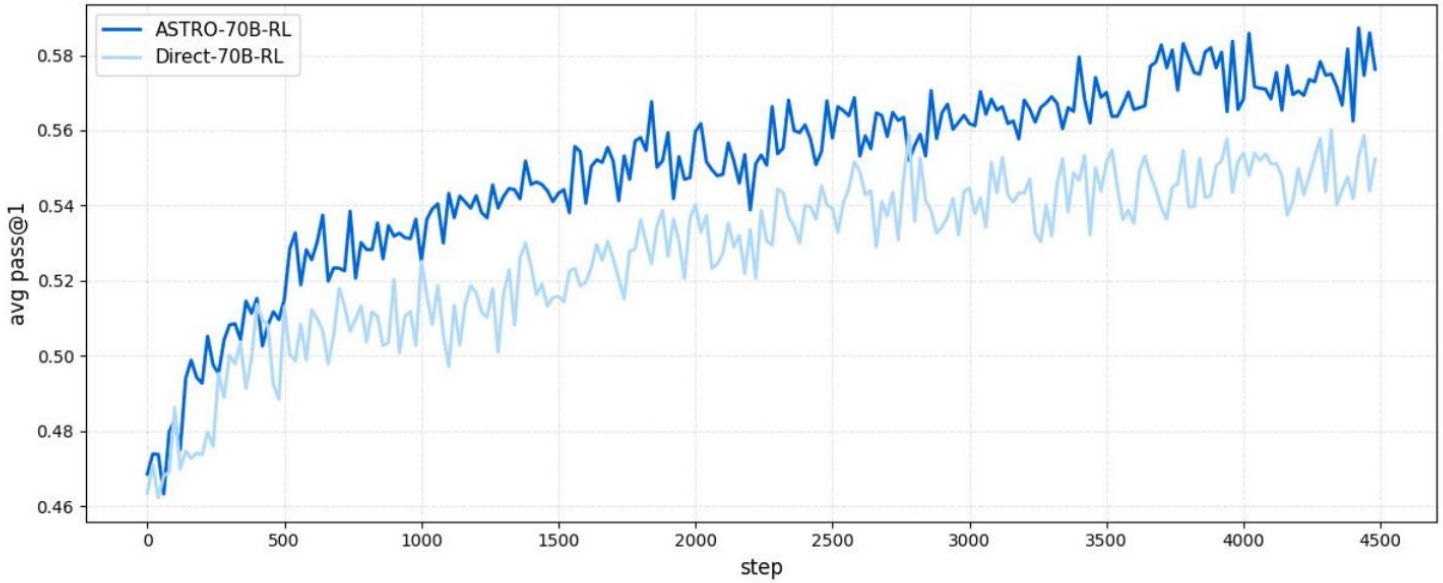
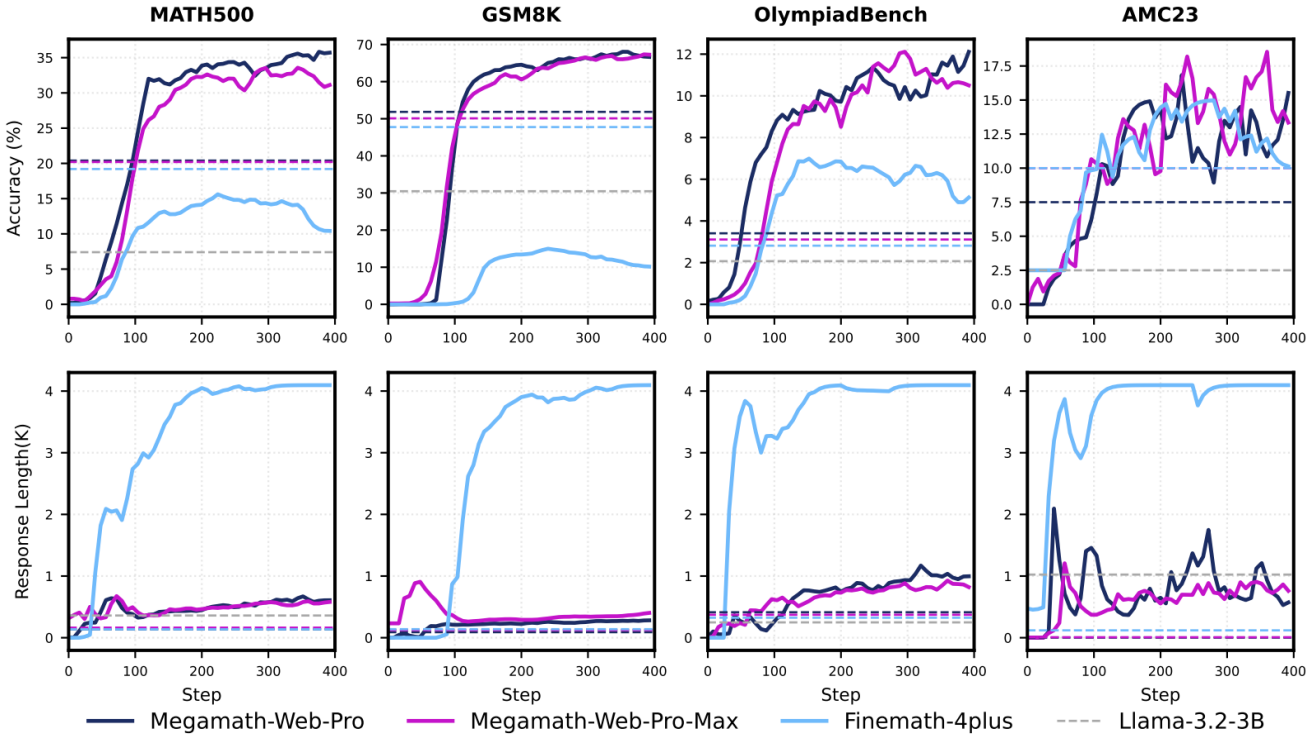


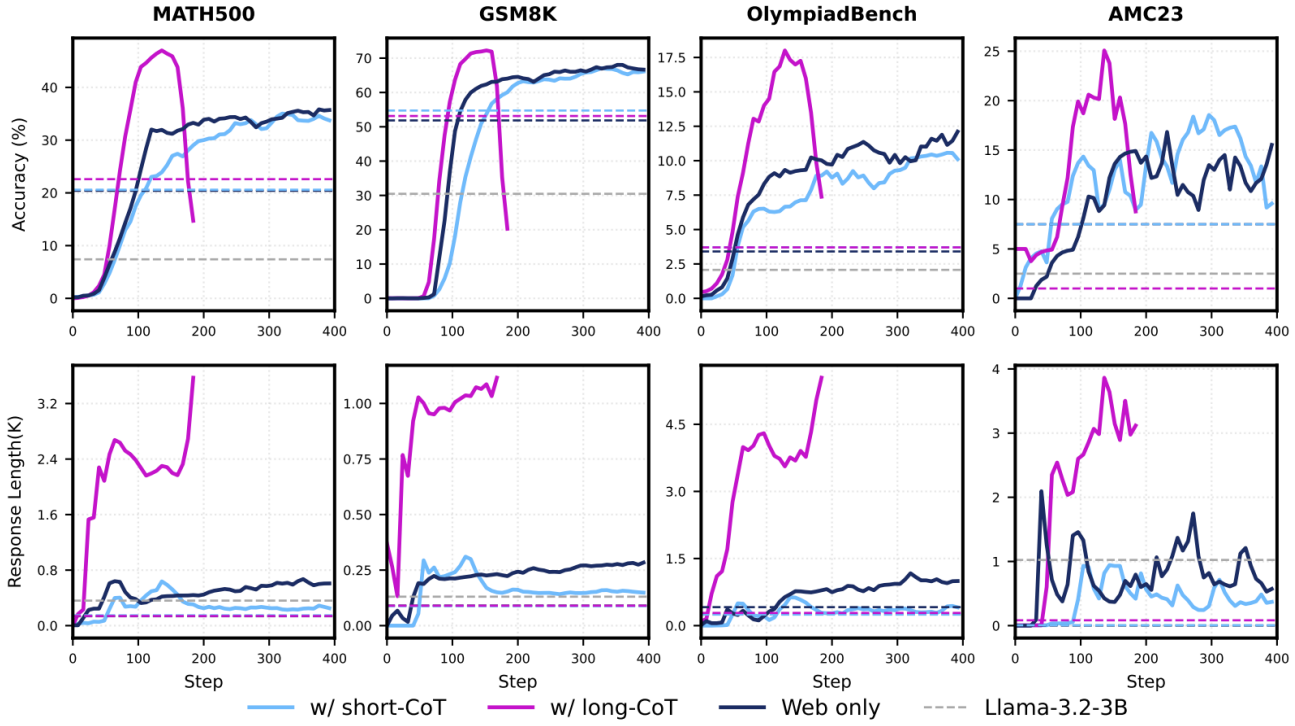
Figure 7 RL training curves for our no-search baseline vs. ASTRO, averaged over the three evaluation benchmarks. The difference between the policy trained with search priors (ASTRO, **dark blue**) and without search priors (Direct, light blue) is noticeable throughout the RL training process, showcasing the importance of infusing search priors for RL.

What is Needed to Unlock Thinking? (4)

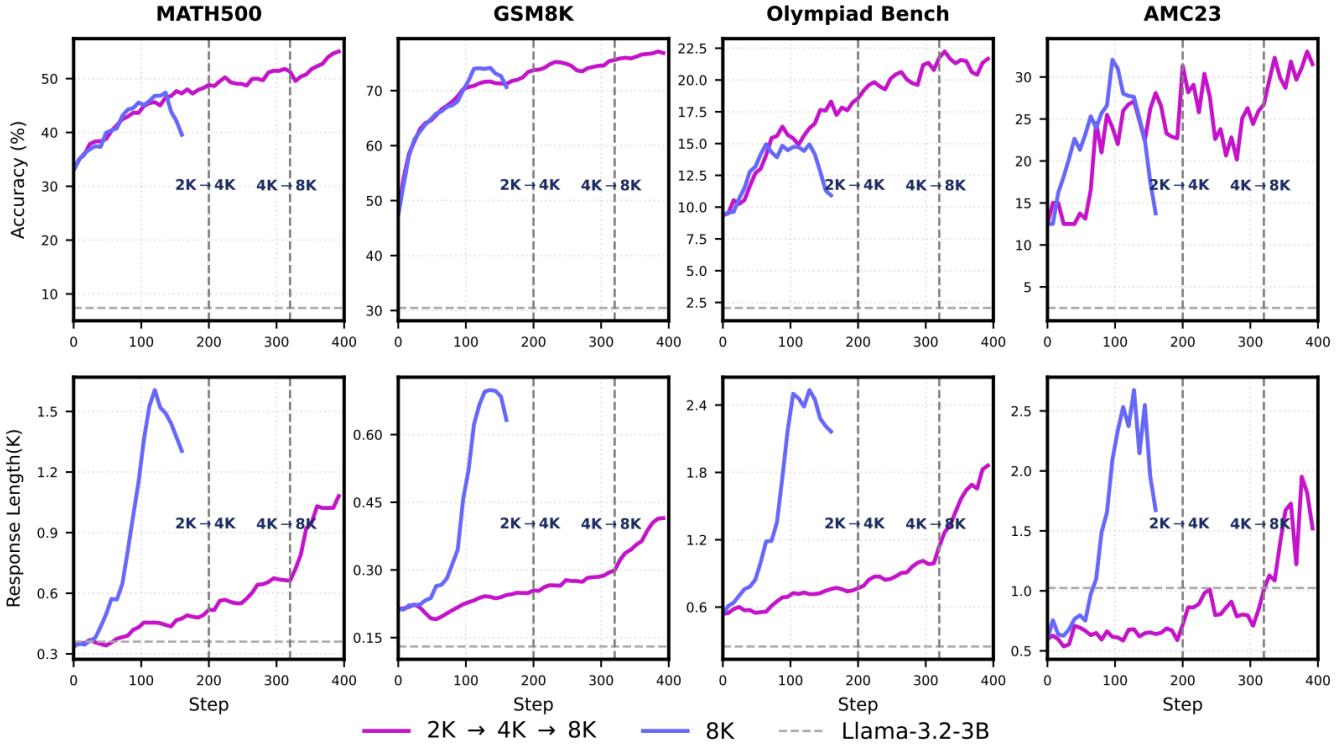


OctoThinker: Mid-training Incentivizes Reinforcement Learning Scaling (Wang et al., 2025)

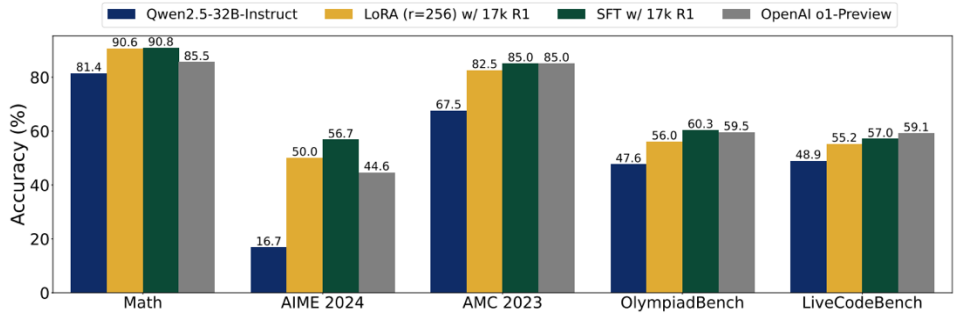
What is Needed to Unlock Thinking? (4)



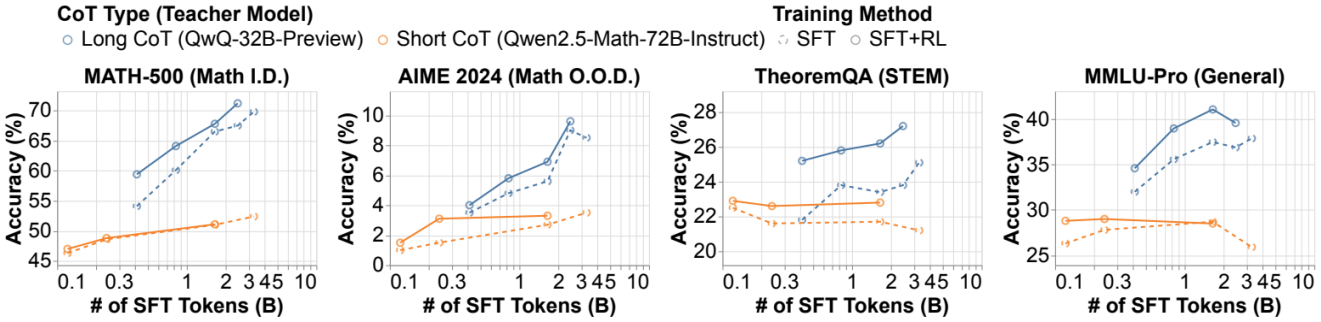
What is Needed to Unlock Thinking? (4)



What is Needed to Unlock Thinking? (4)

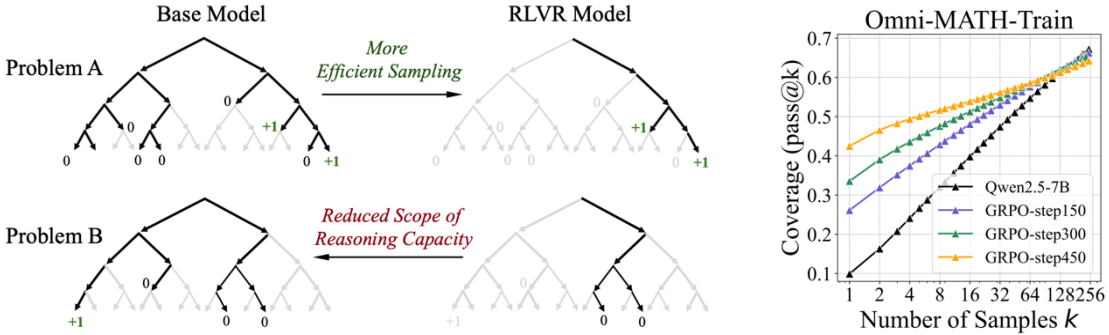


LLMs Can Easily Learn to Reason from Demonstrations Structure, not content, is what matters! (Li et al., 2025)

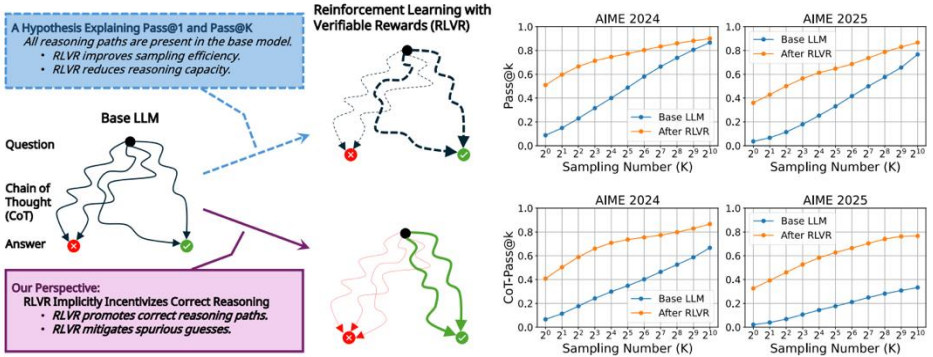


Demystifying Long Chain-of-Thought Reasoning in LLMs (Yeo et al., 2025)

Does RL really improve intelligence?

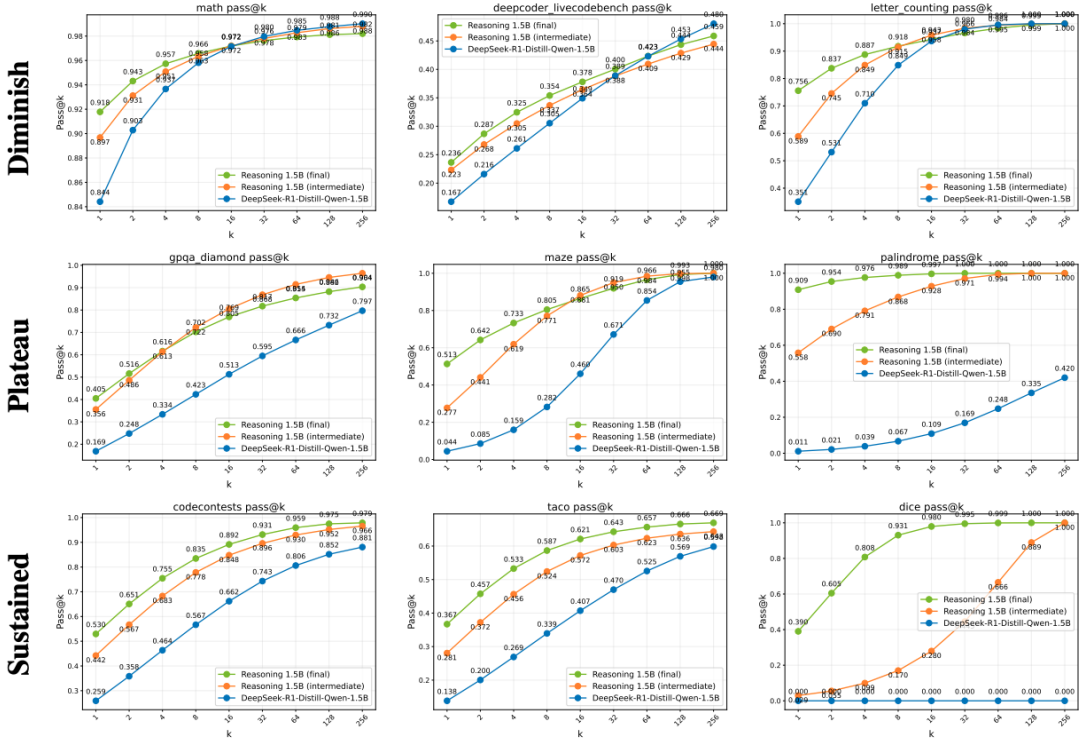


Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? (Yue et al., 2025)



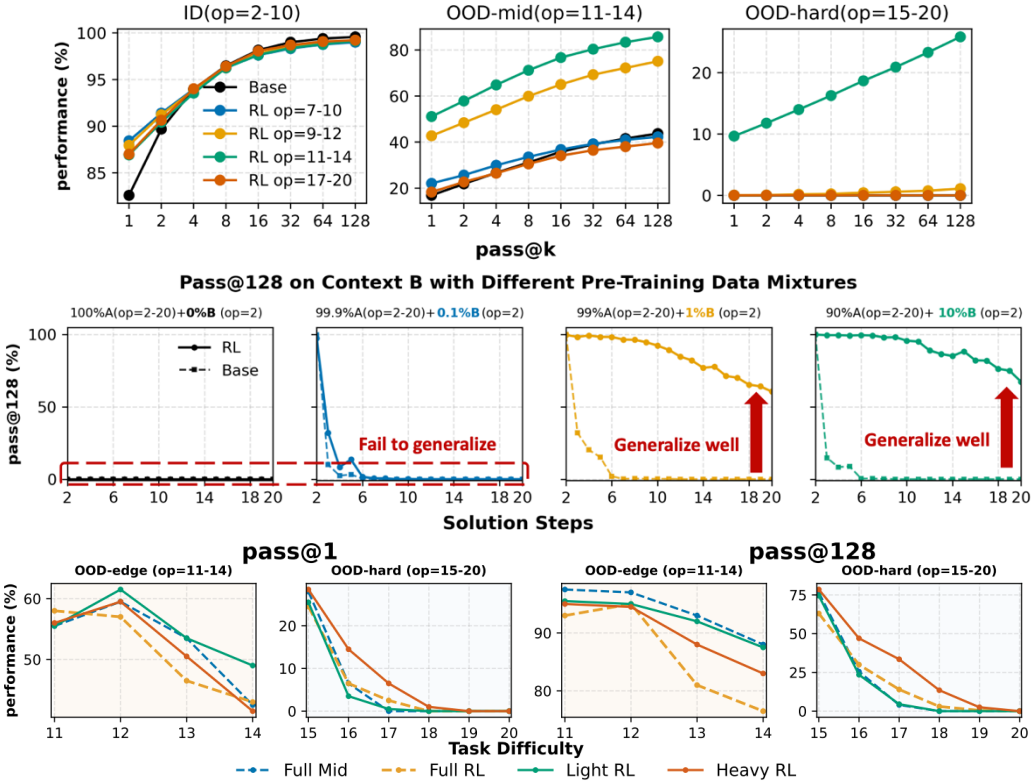
Reinforcement Learning with Verifiable Rewards Implicitly Incentivizes Correct Reasoning in Base LLMs (Wen et al., 2025)

Does RL really improve intelligence?



ProRL: Prolonged Reinforcement Learning Expands Reasoning Boundaries in Large Language Models (Liu et al., 2025)

Does RL really improve intelligence?



On the Interplay of Pre-Training, Mid-Training, and RL on Reasoning Language Models (Zhang et al., 2025)

Does RL really improve intelligence?

Takeaway 1

RL produces true capability gains (`pass@128`) beyond base models only when two conditions hold: (i) The task is not heavily covered during pre-training, leaving sufficient headroom for exploration; and (ii) the RL data is calibrated to the model's *edge of competence*, neither too easy (in-distribution) nor too hard (out-of-distribution).

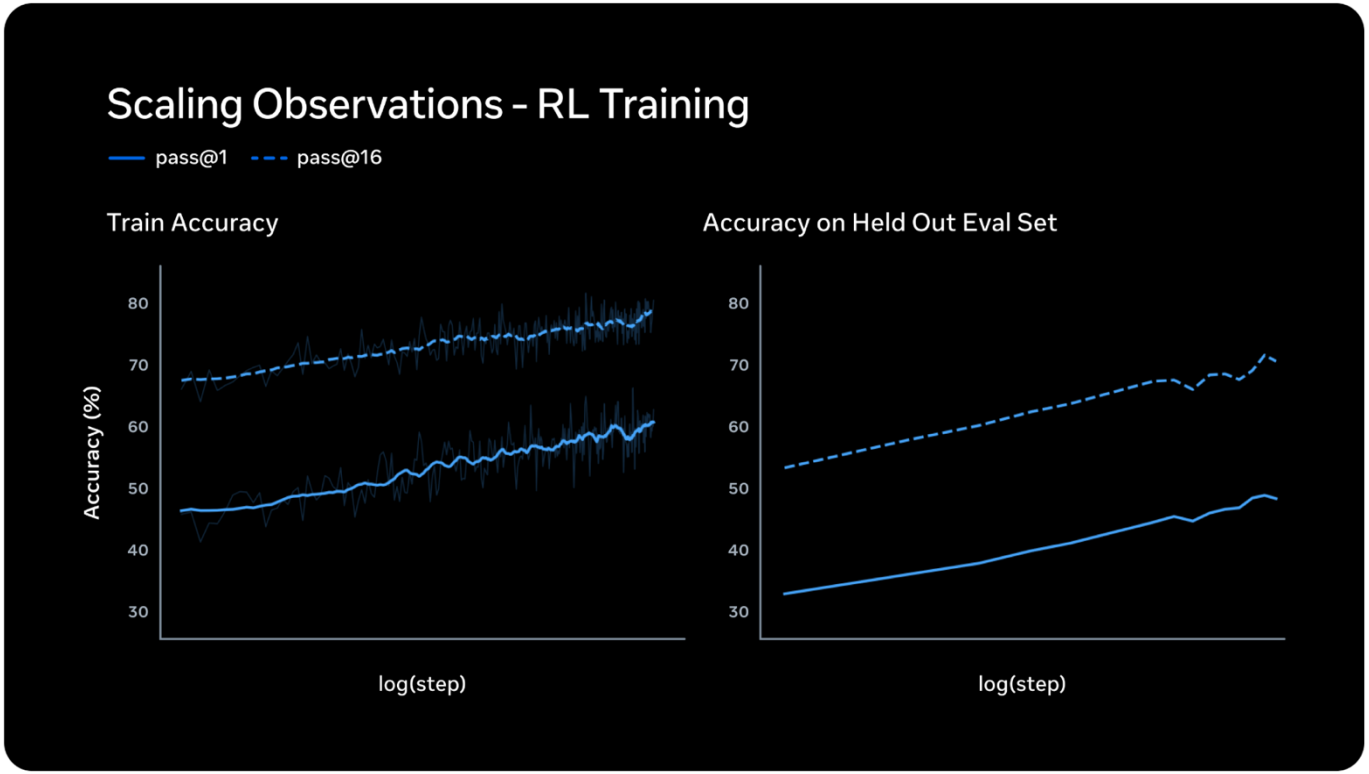
Takeaway 2

RL incentivizes contextual generalization only when the base model already contains the necessary primitives. Without minimal pre-training exposure to a new context, RL cannot induce transfer. However, even sparse exposure (e.g., $\geq 1\%$) provides a sufficient seed that RL can reinforce during post-training, yielding robust cross-context generalization.

Takeaway 3

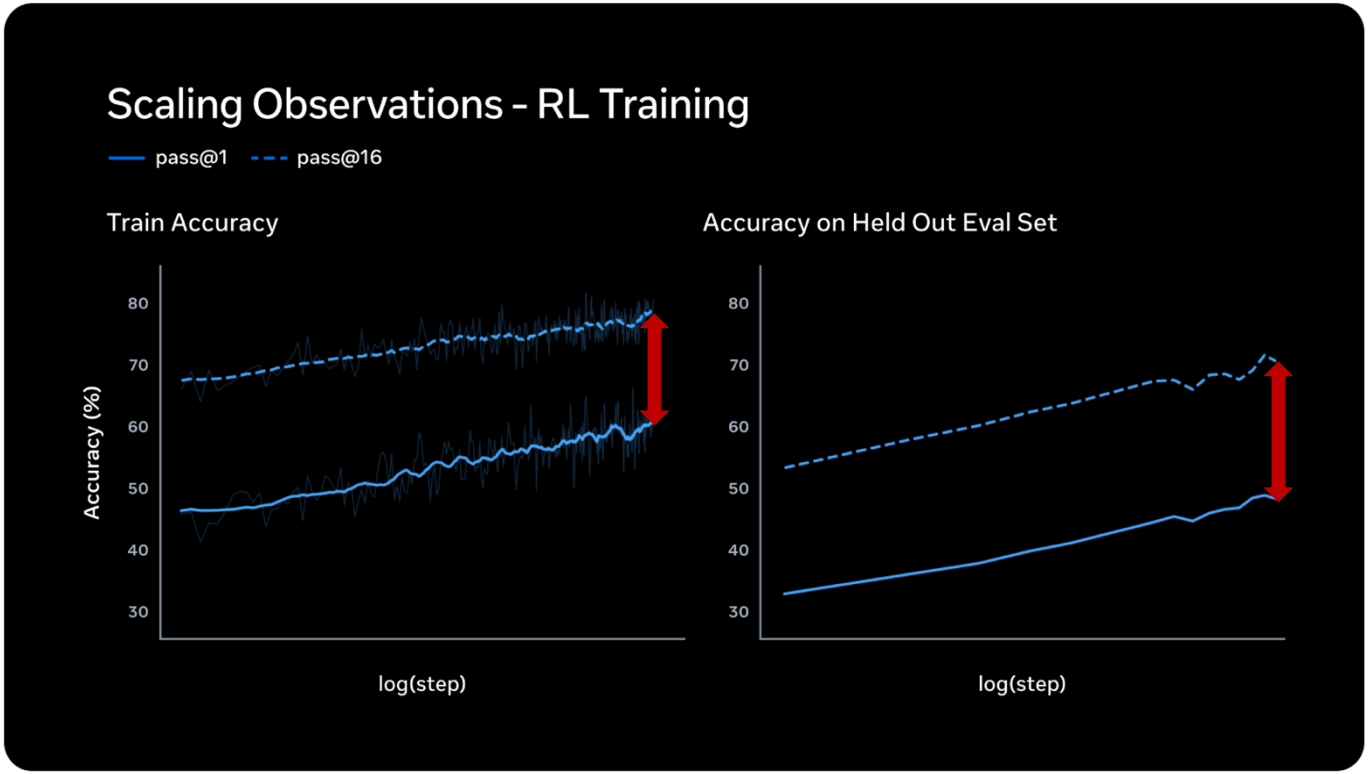
Introducing a mid-training phase that bridges pre- and post-training distributions substantially strengthens generalization under a fixed compute budget. This highlights mid-training as an underexplored but powerful lever in training design. Compute should be allocated in a task-aware manner: (i) when prioritizing in-distribution performance, allocate more budget to mid-training with only light RL; (ii) for out-of-distribution generalization, reserve a modest portion of compute for mid-training to establish essential priors, and dedicate the remaining budget to heavier RL exploration.

Does RL really improve intelligence?



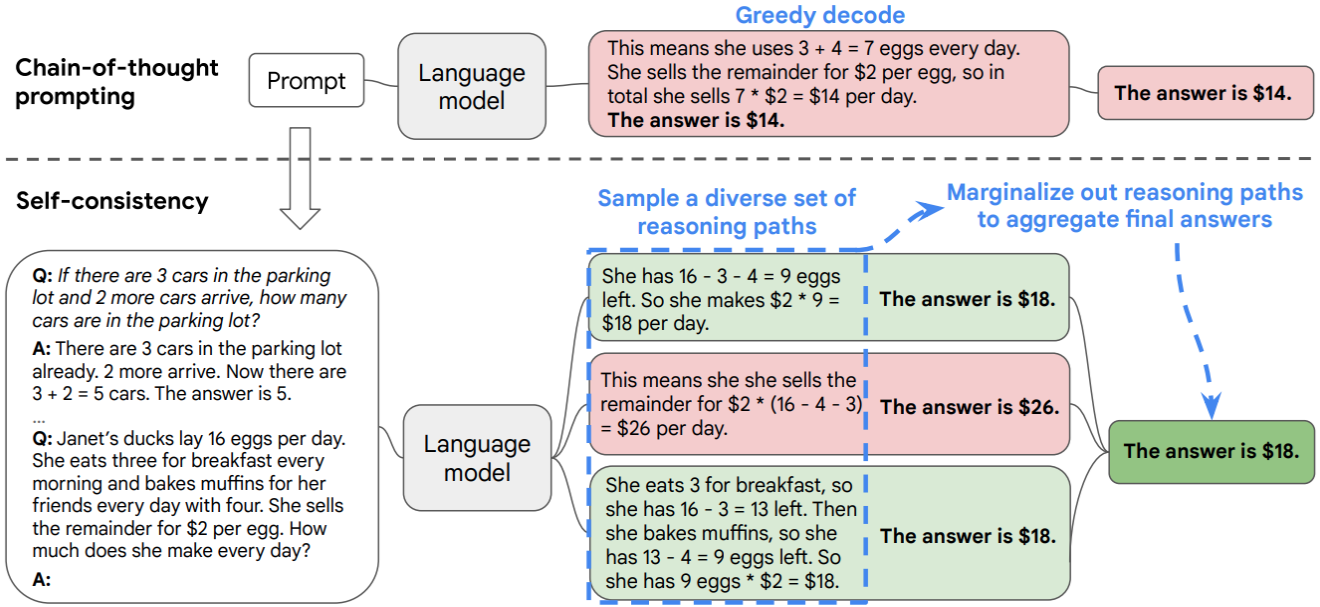
https://ai.meta.com/blog/introducing-muse-spark-msl/ (Meta, 2026)

Can we squeeze more capabilities?



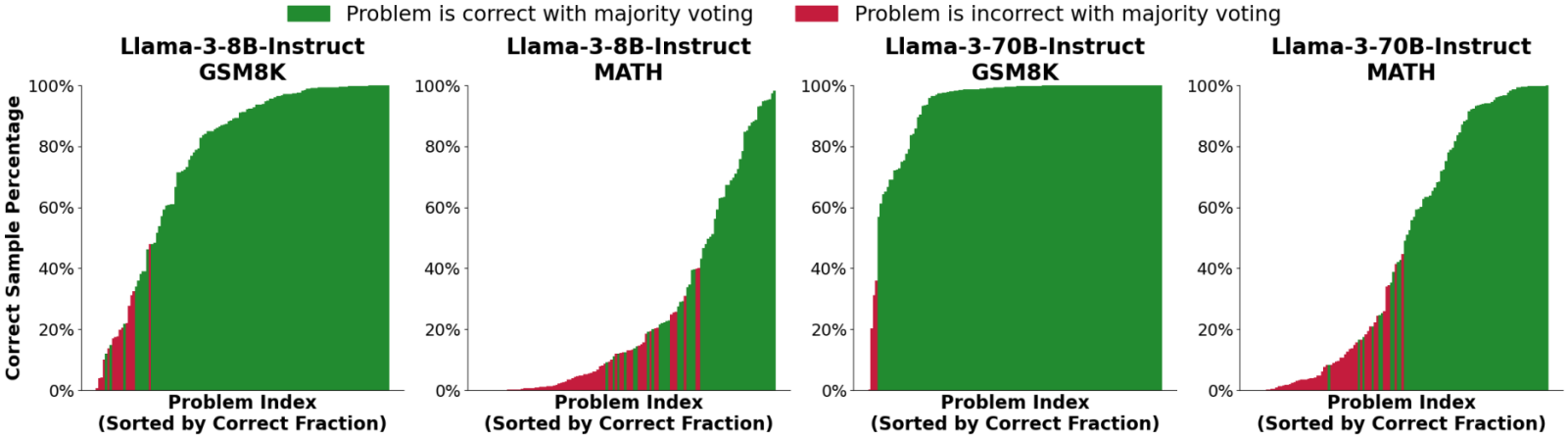
https://ai.meta.com/blog/introducing-muse-spark-msl/ (Meta, 2026)

Aggregate across parallel generations!!



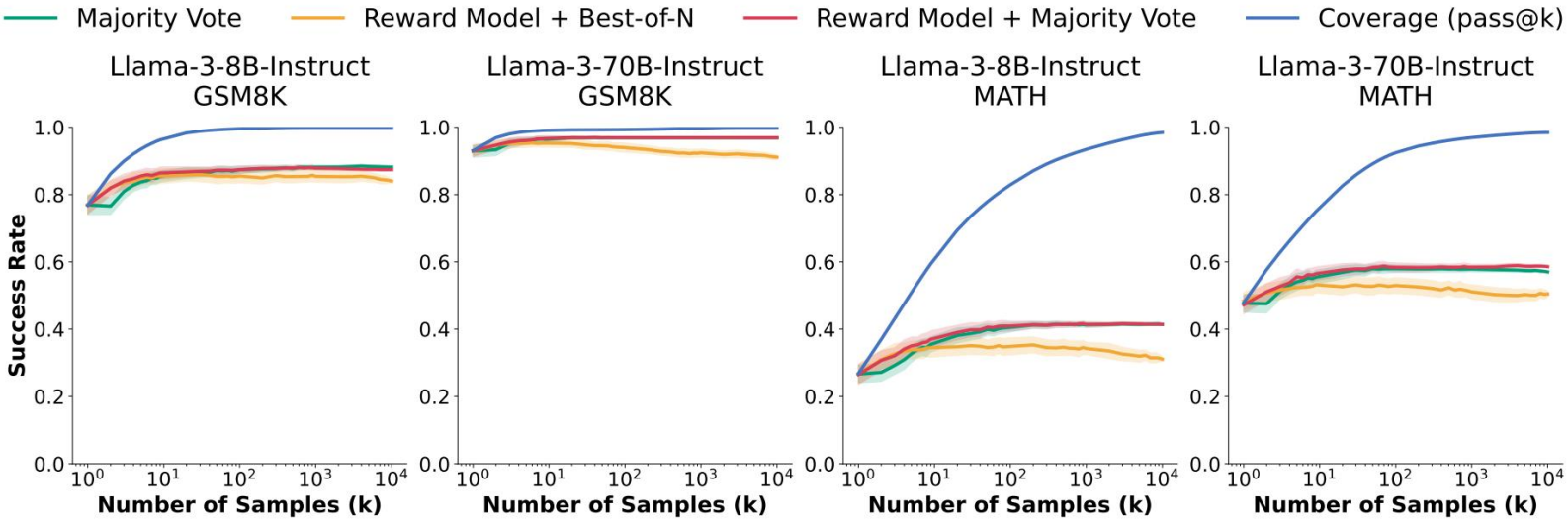
Self-Consistency Improves Chain of Thought Reasoning in Language Models (Google DeepMind, 2023)

Majority Voting does not always work



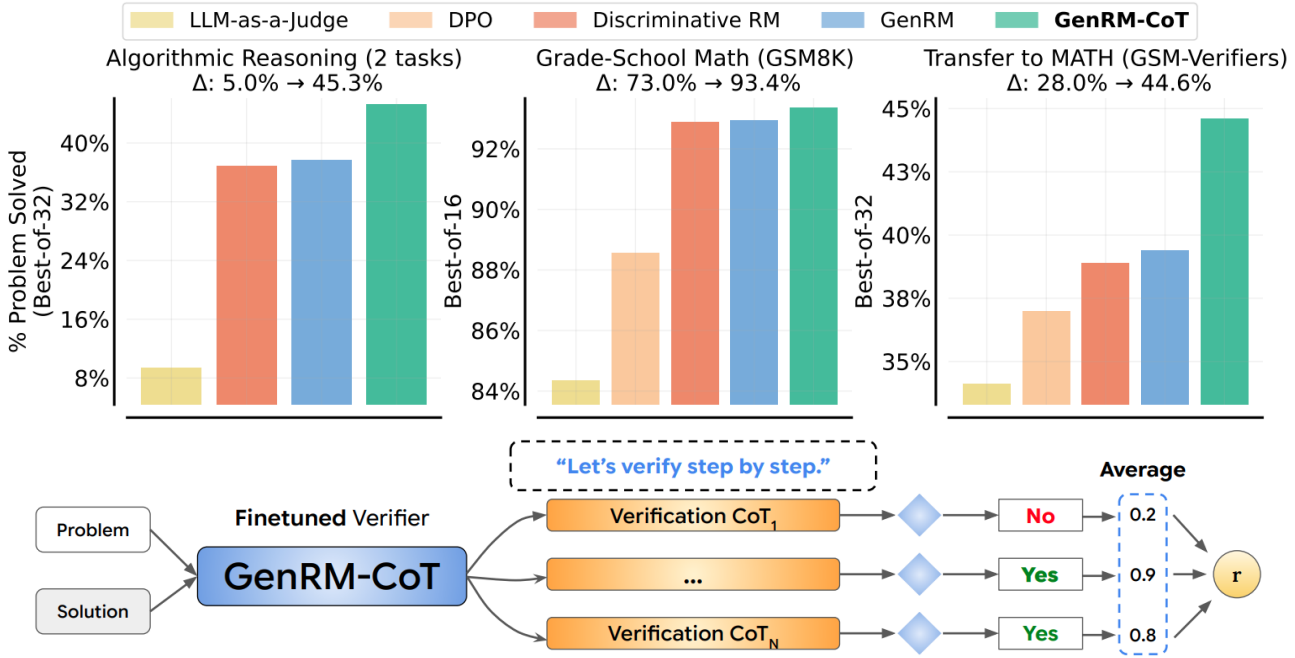
Large Language Monkeys: Scaling Inference Compute with Repeated Sampling (Brown et al., 2024)

Verification is not a trivial task



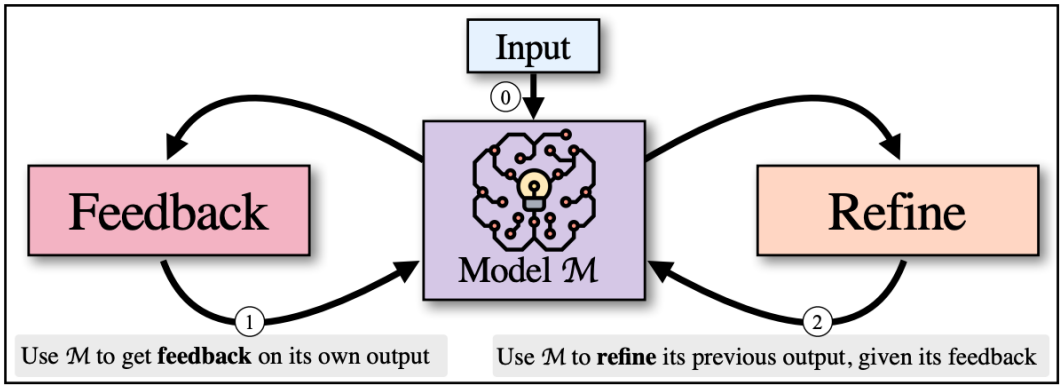
Large Language Monkeys: Scaling Inference Compute with Repeated Sampling (Brown et al., 2024)

Training for Verification



Generative Verifiers: Reward Modeling as Next-Token Prediction (Google DeepMind, 2025)

Another Dimension – Sequential Refinement



(d) **Code optimization:** x, $\mathbf{y_t}$

```
Generate sum of 1, ..., N
def sum(n):
    res = 0
    for i in range(n+1):
        res += i
    return res
```

(e) FEEDBACK **fb**

```
This code is slow as
it uses brute force.
A better approach is
to use the formula
... (n(n+1))/2.
```

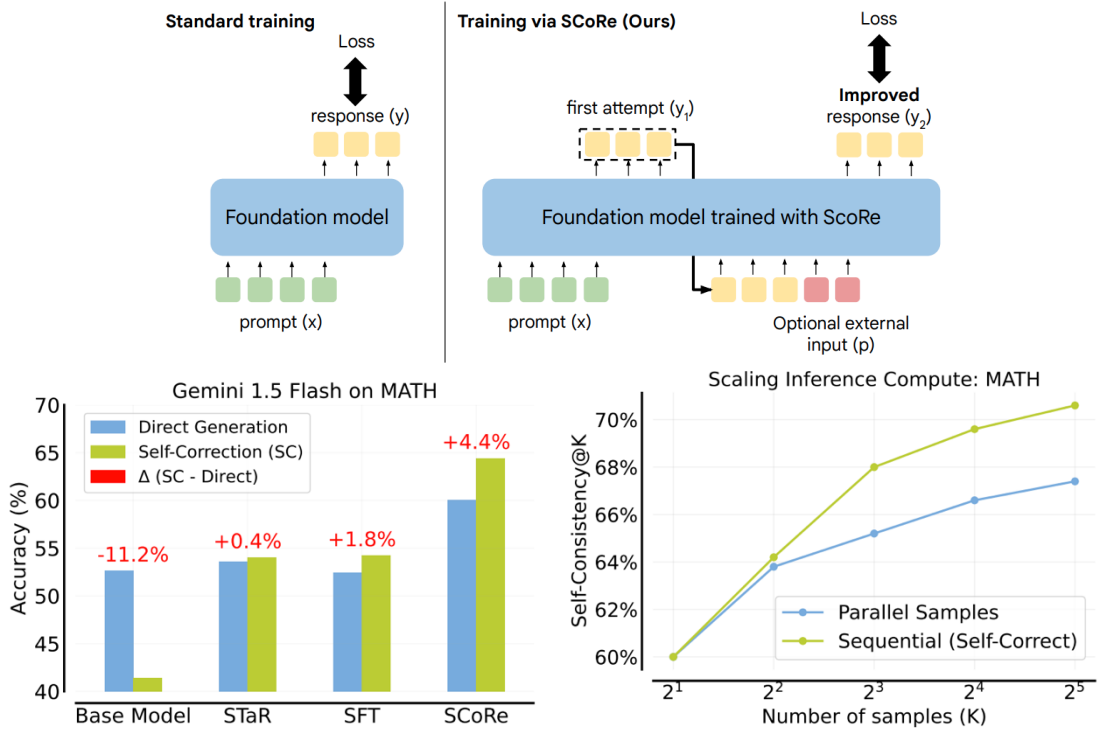
(f) REFINE $\mathbf{y_{t+1}}$

```
Code (refined)

def sum_faster(n):
  return (n*(n+1))//2
```

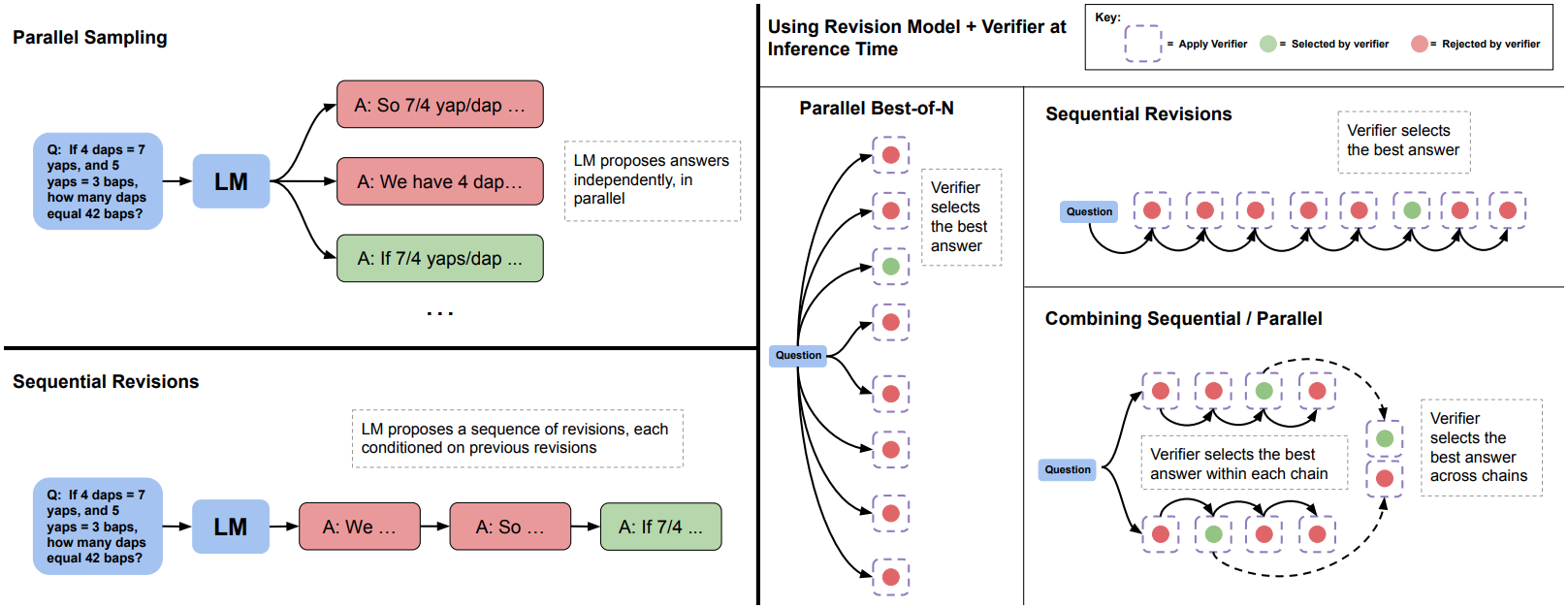
Self-Refine: Iterative Refinement with Self-Feedback (Madaan et al., 2023)

Training for Self-Correction



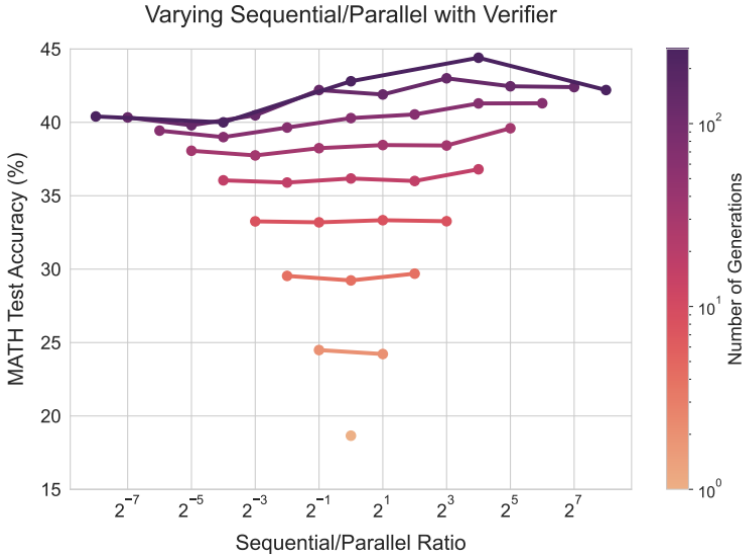
Training Language Models to Self-Correct via Reinforcement Learning (Google DeepMind, 2024)

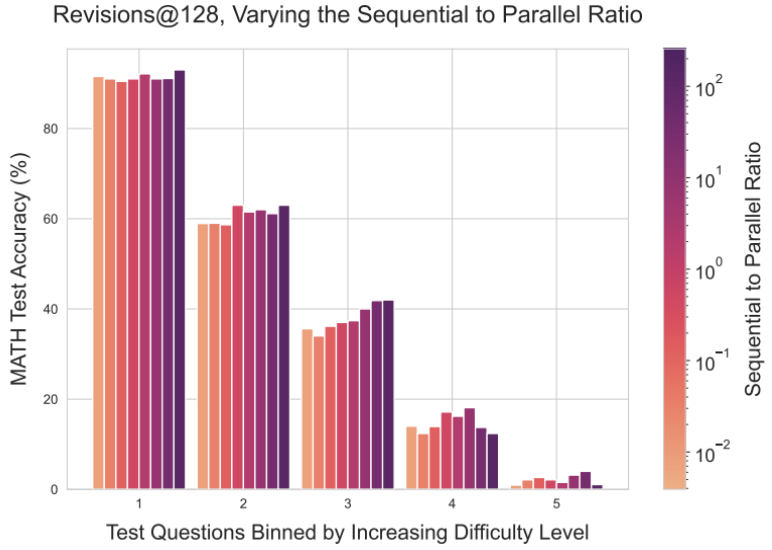
Putting it Together…



Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters (Snell et al., 2024)

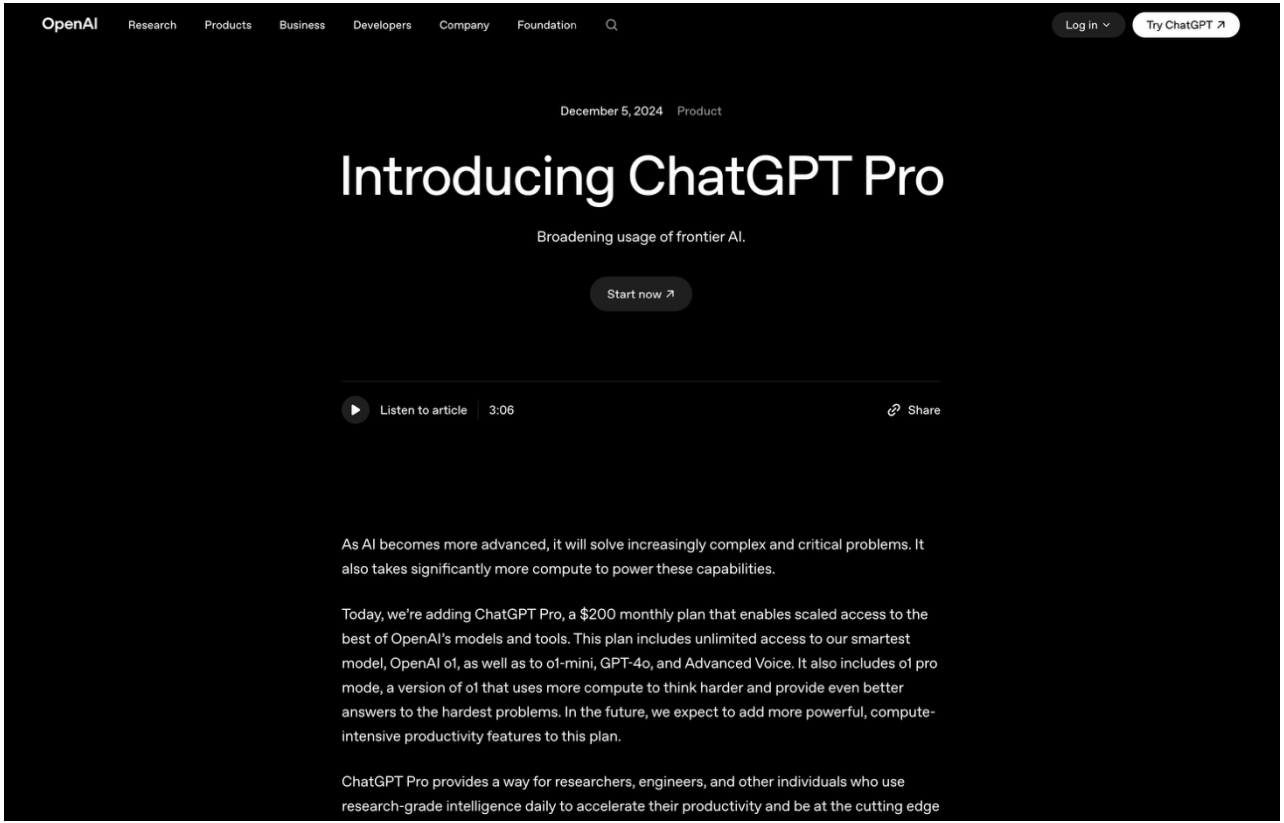
Mix Parallel and Sequential Dimensions





Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters (Snell et al., 2024)

Scaling Test-Time Reasoning



Scaling Test-Time Reasoning

Gemini 3 Deep Think: Advancing science, research and engineering

Feb 12, 2026
4 min read

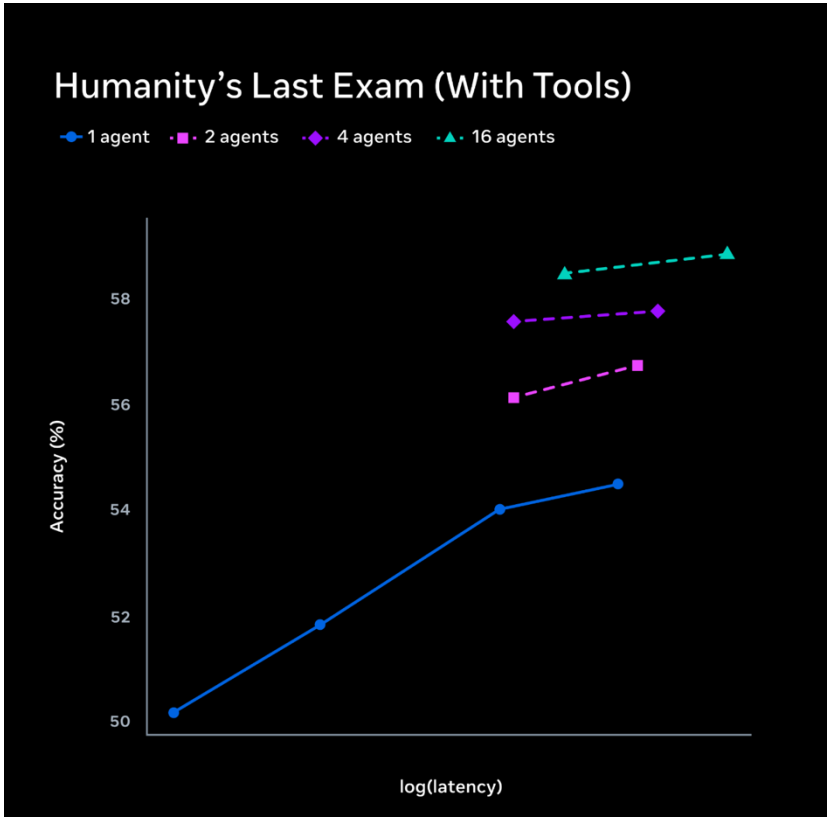
Our most specialized reasoning mode is now updated to solve modern science, research and engineering challenges.

The Deep Think team

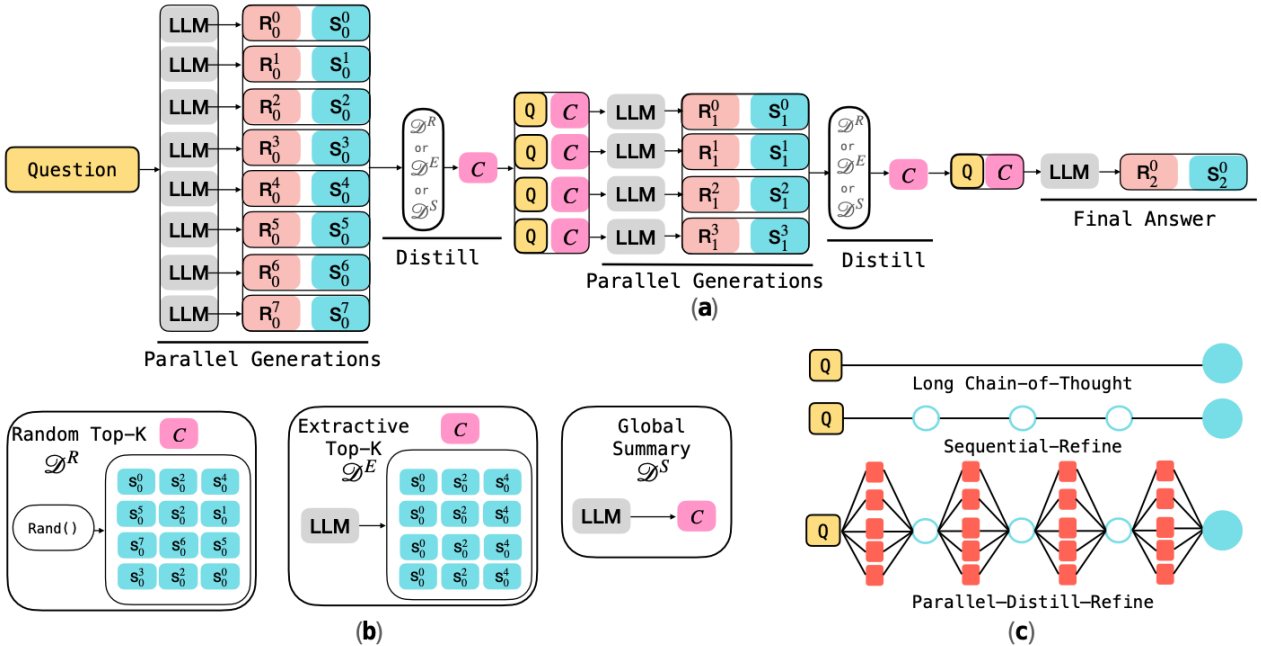
Read AI-generated summary Share

Scaling Test-Time Reasoning

Benchmark	Muse Spark Contemplating	Gemini 3.1 Deep Think	GPT 5.4 Pro
Humanity's Last Exam Multidisciplinary Reasoning (No Tools)	50.2	48.4	43.9 Self-Reported: 42.7
Humanity's Last Exam Multidisciplinary Reasoning (With Tools)	58.4	53.4	58.7
IPhO 2025 (Theory) Physics Olympiad	82.6	87.7	93.5
FrontierScience Research Scientific Research	38.3	23.3*	36.7

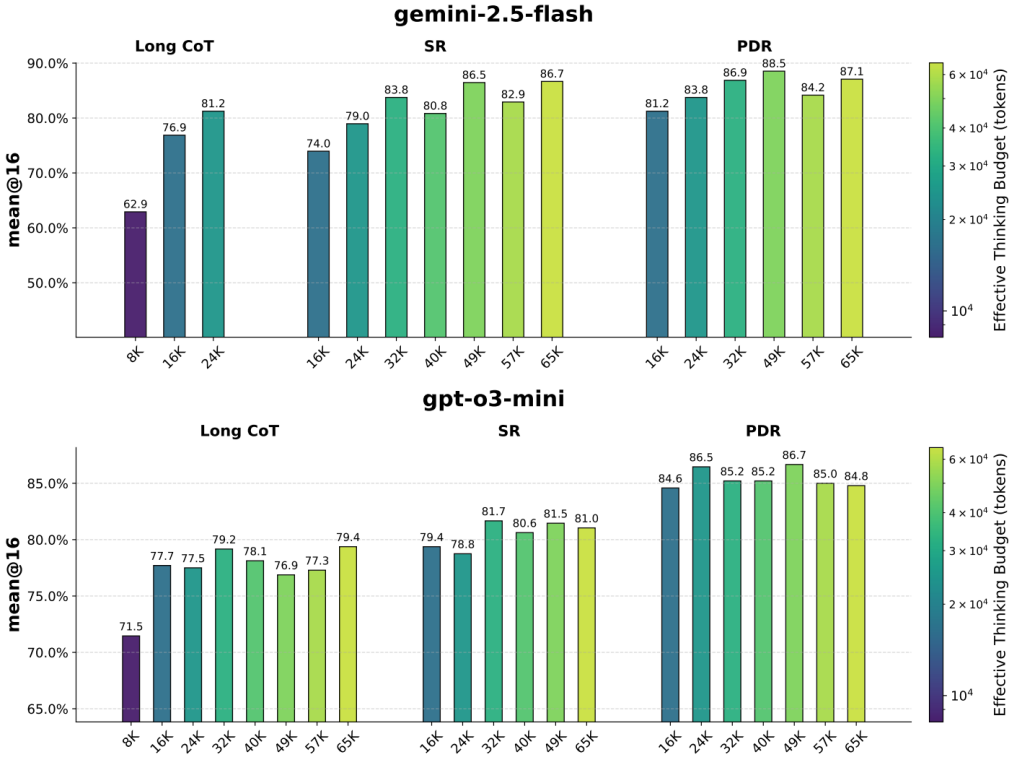


Parallel Aggregation + Sequential Refinement

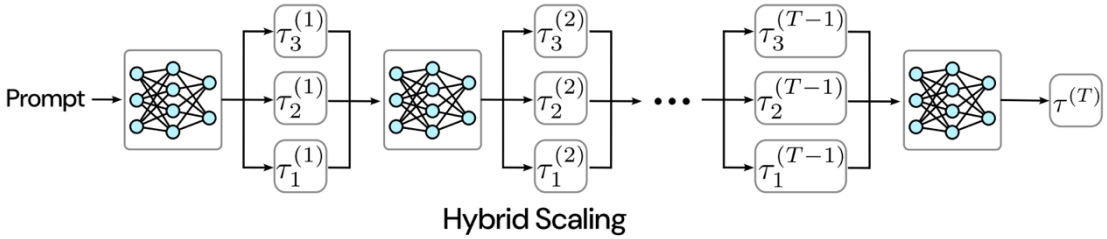


Rethinking Thinking Tokens: LLMs as Improvement Operators (Meta, 2025)

Parallel Aggregation + Sequential Refinement

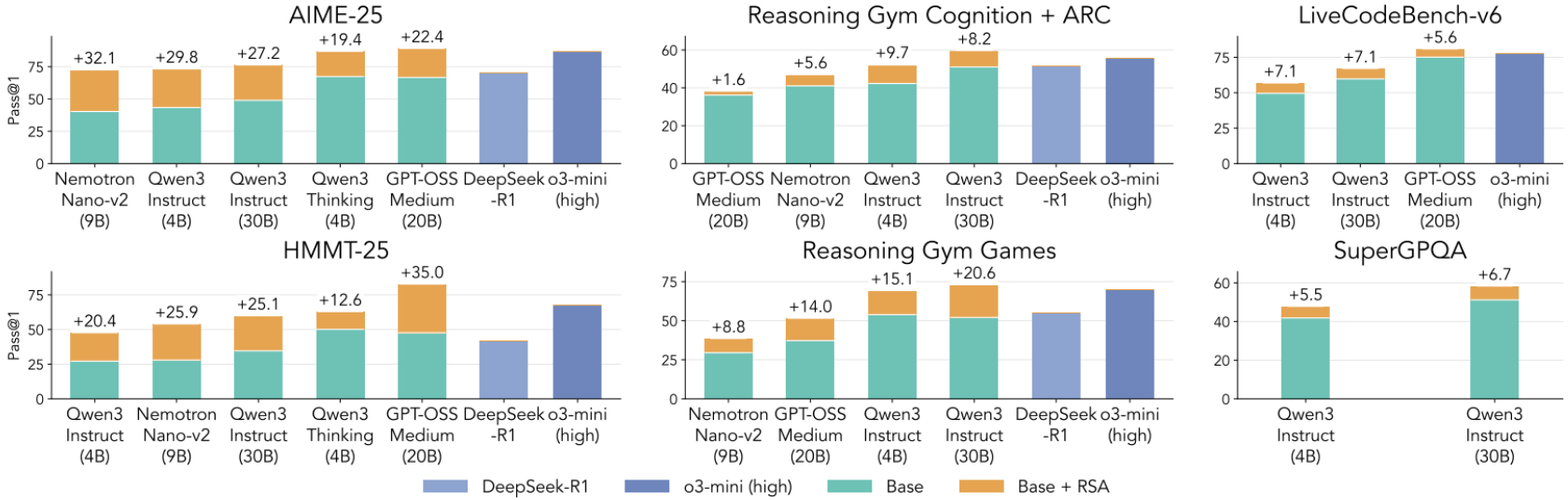


Parallel Aggregation + Sequential Refinement

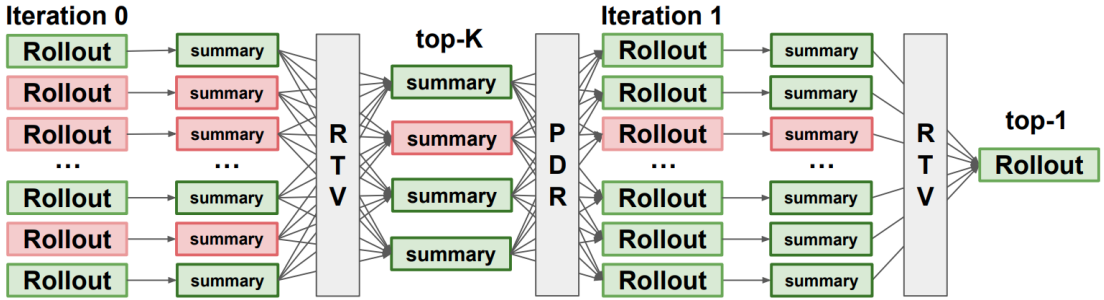


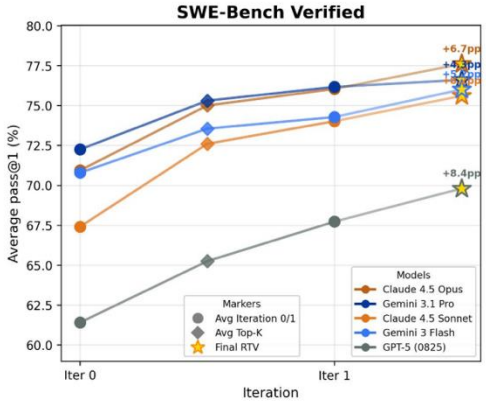
Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models (Venkatraman et al., 2025)

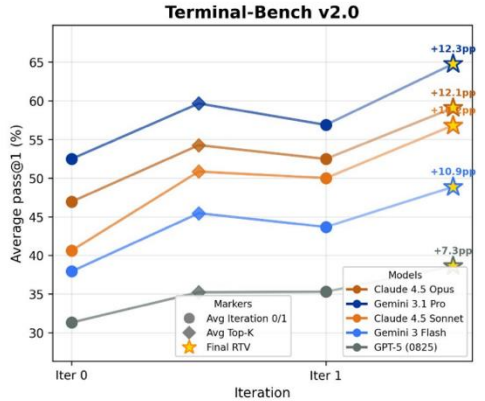
Parallel Aggregation + Sequential Refinement



Agentic Test-time Scaling

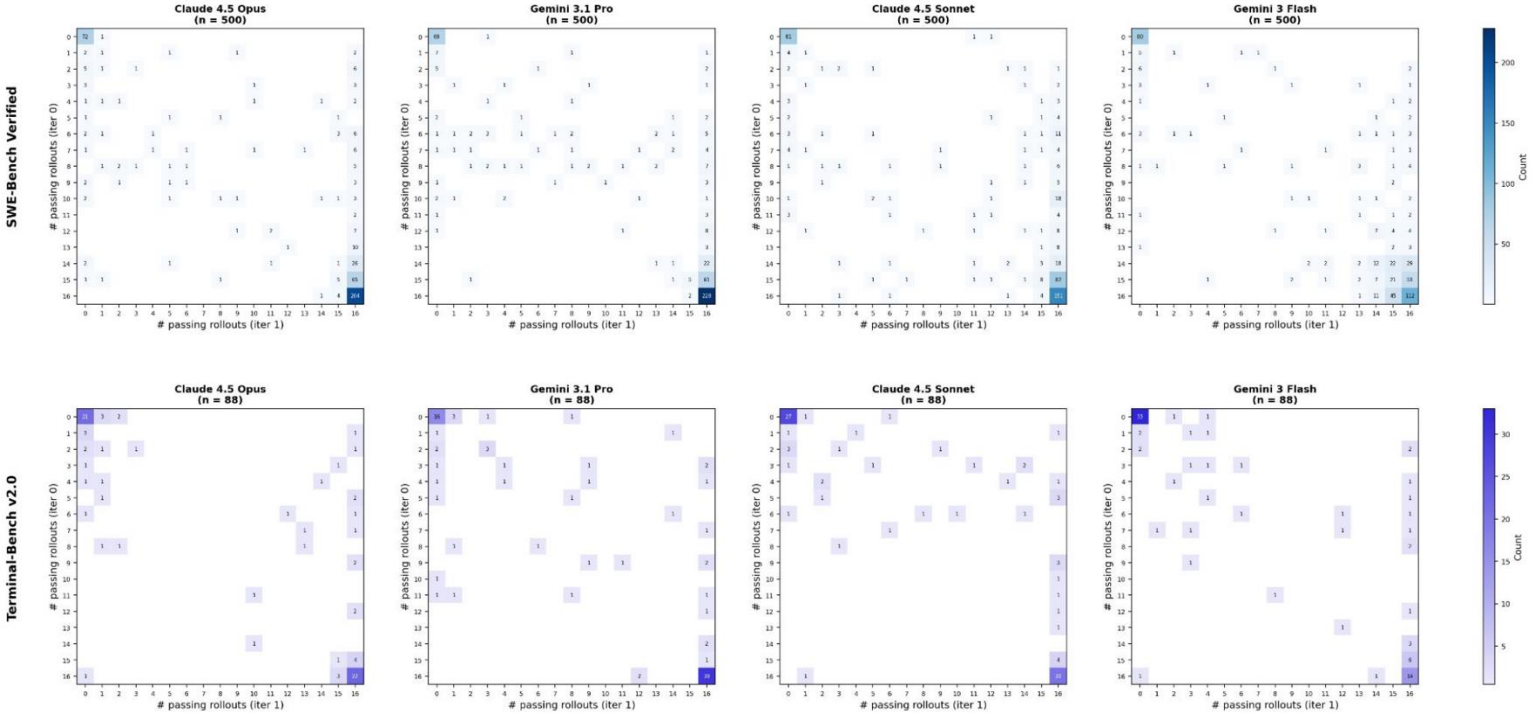






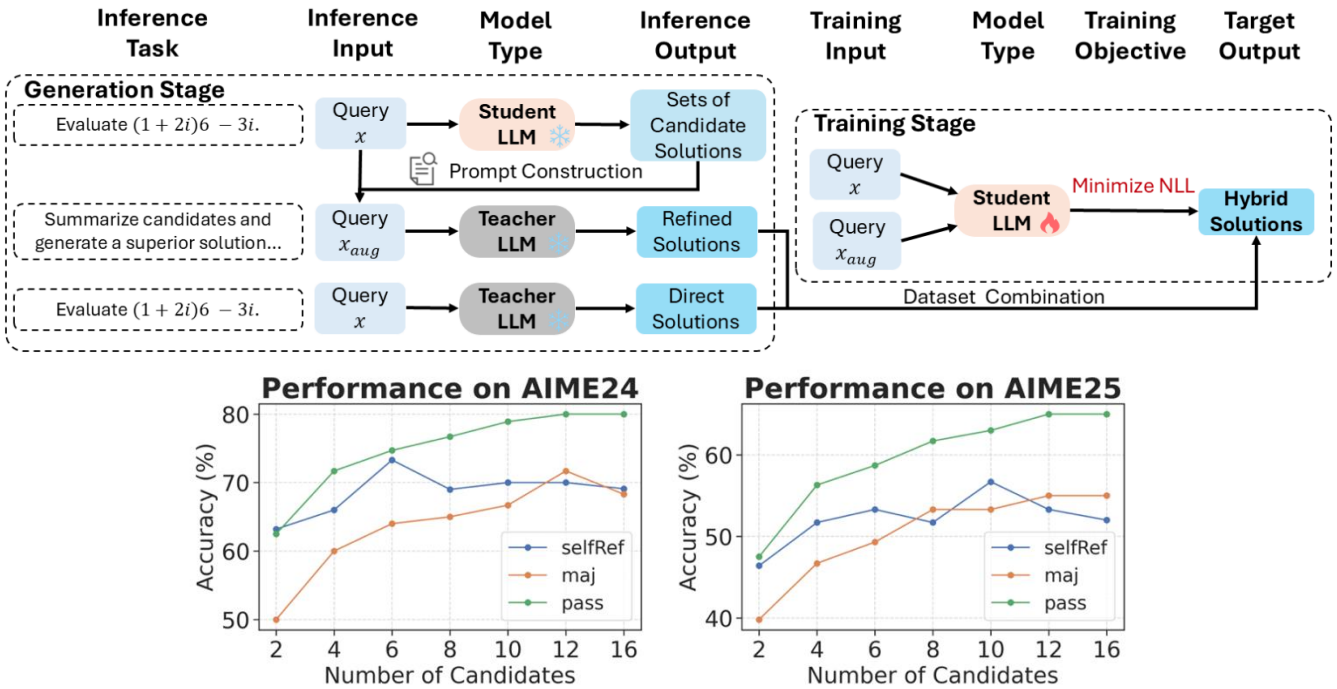
Scaling Test-Time Compute for Agentic Coding (Kim et al., 2026)

Agentic Test-time Scaling



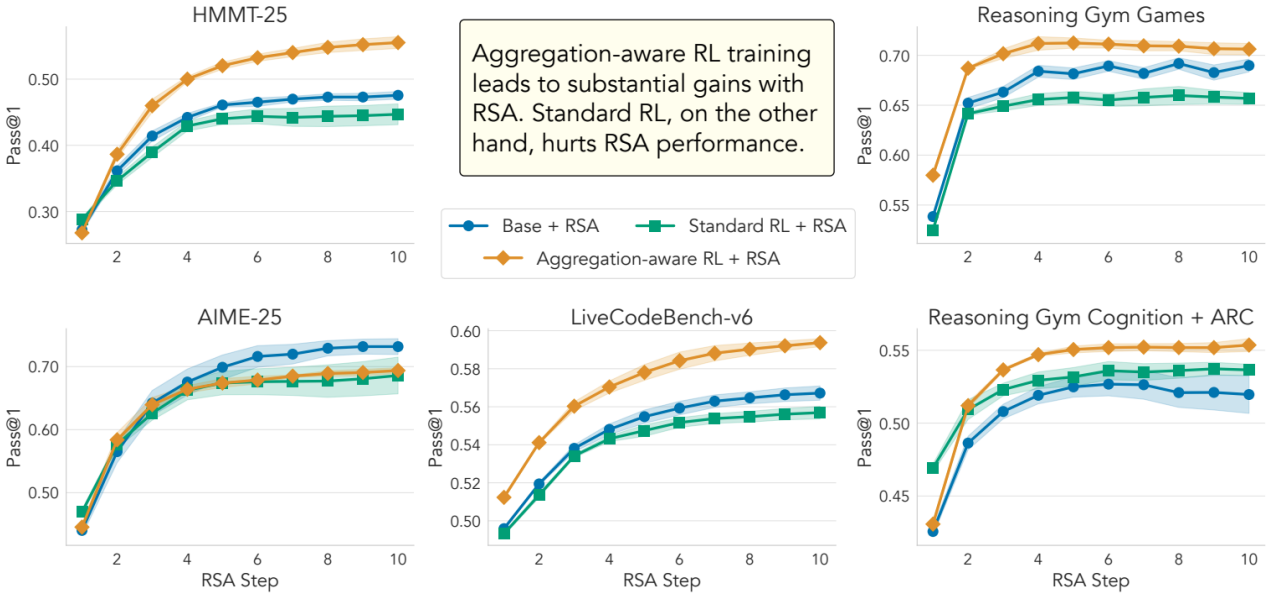
Scaling Test-Time Compute for Agentic Coding (Kim et al., 2026)

Supervised Fine-Tuning on Complex Scaffolds



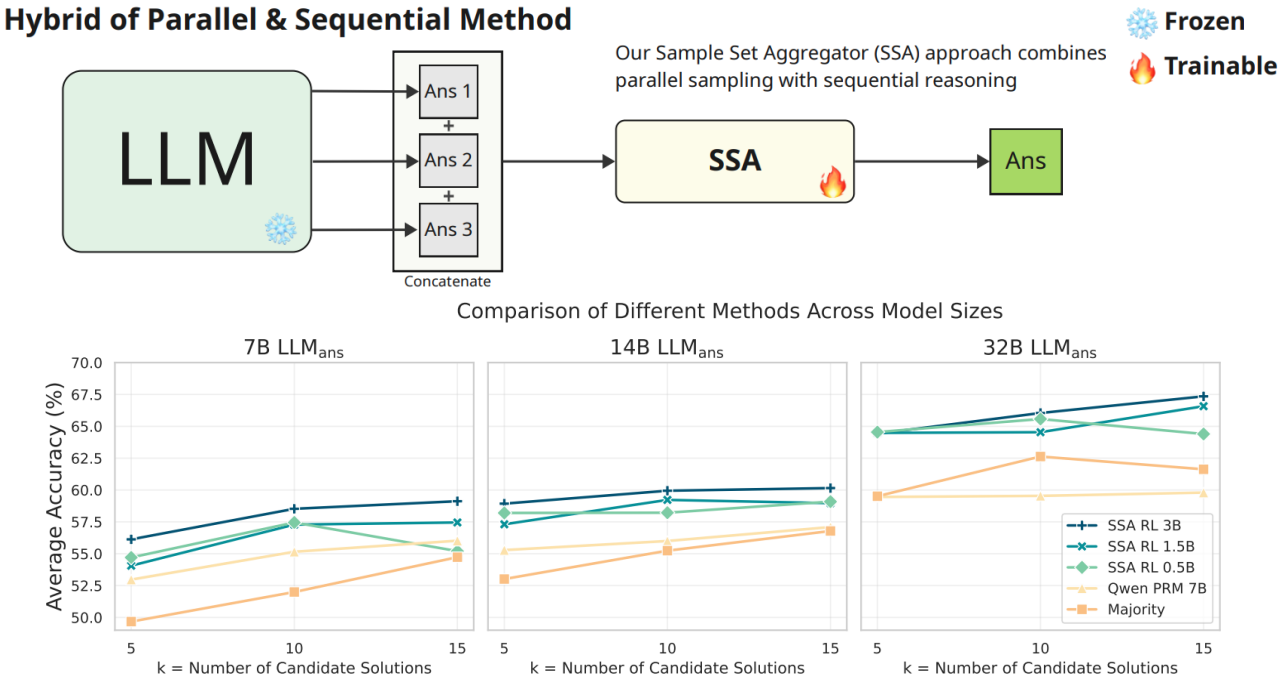
Learning to Refine: Self-Refinement of Parallel Reasoning in LLMs (Wang et al., 2025)

Reinforcement Learning on Complex Scaffolds



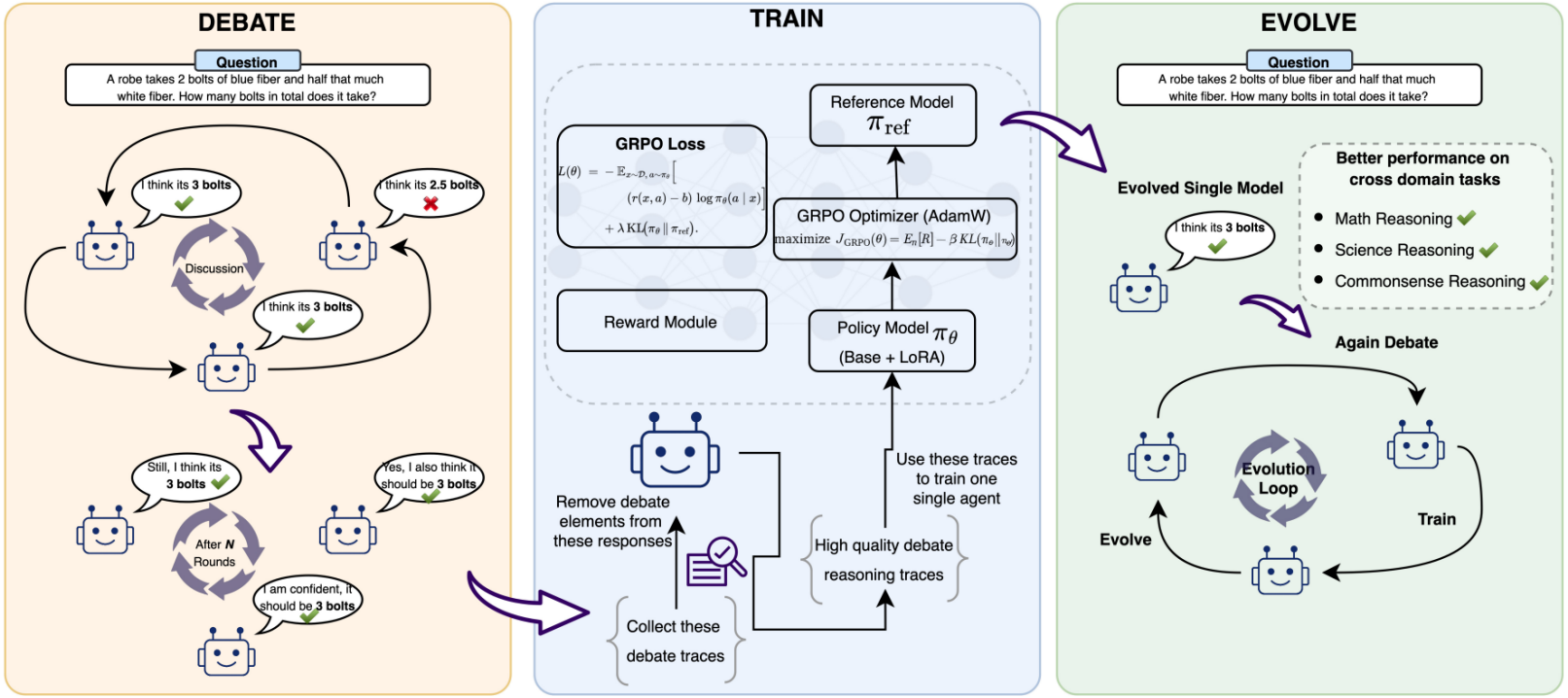
Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models *(Venkatraman et al., 2025)*

Reinforcement Learning on Complex Scaffolds



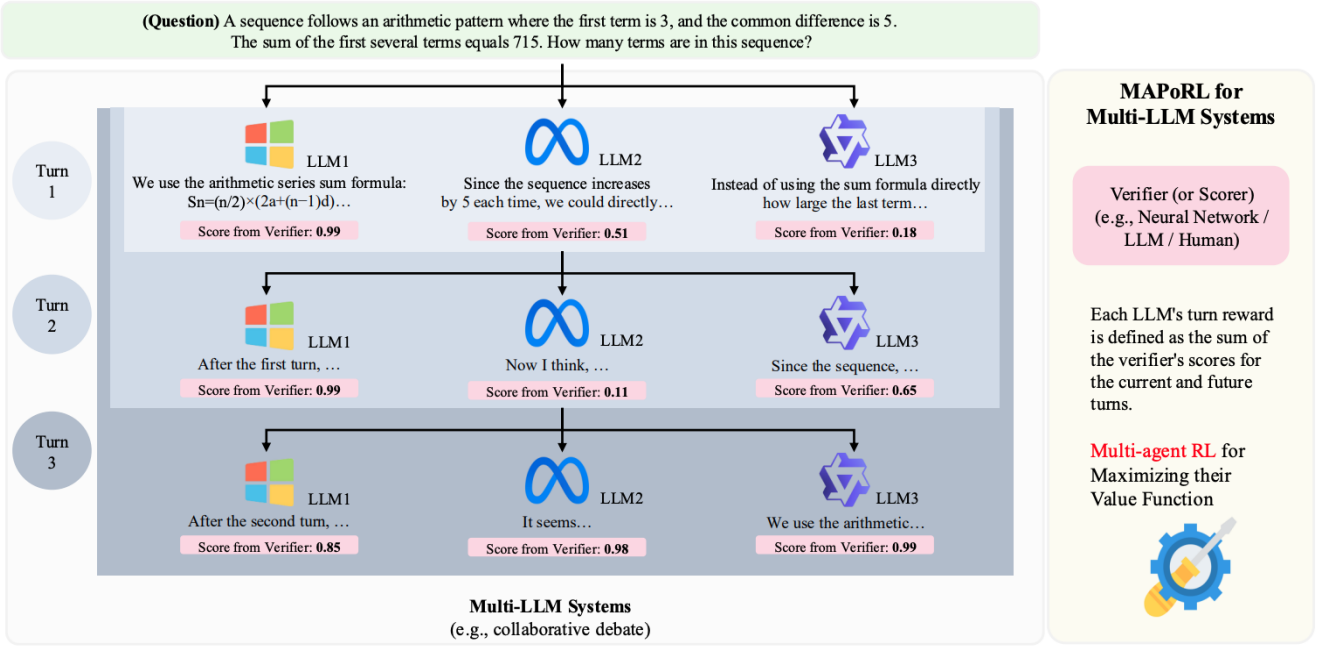
Learning to Reason Across Parallel Samples for LLM Reasoning (Qi et al., 2025)

RL on Multi-Agent Scaffolds



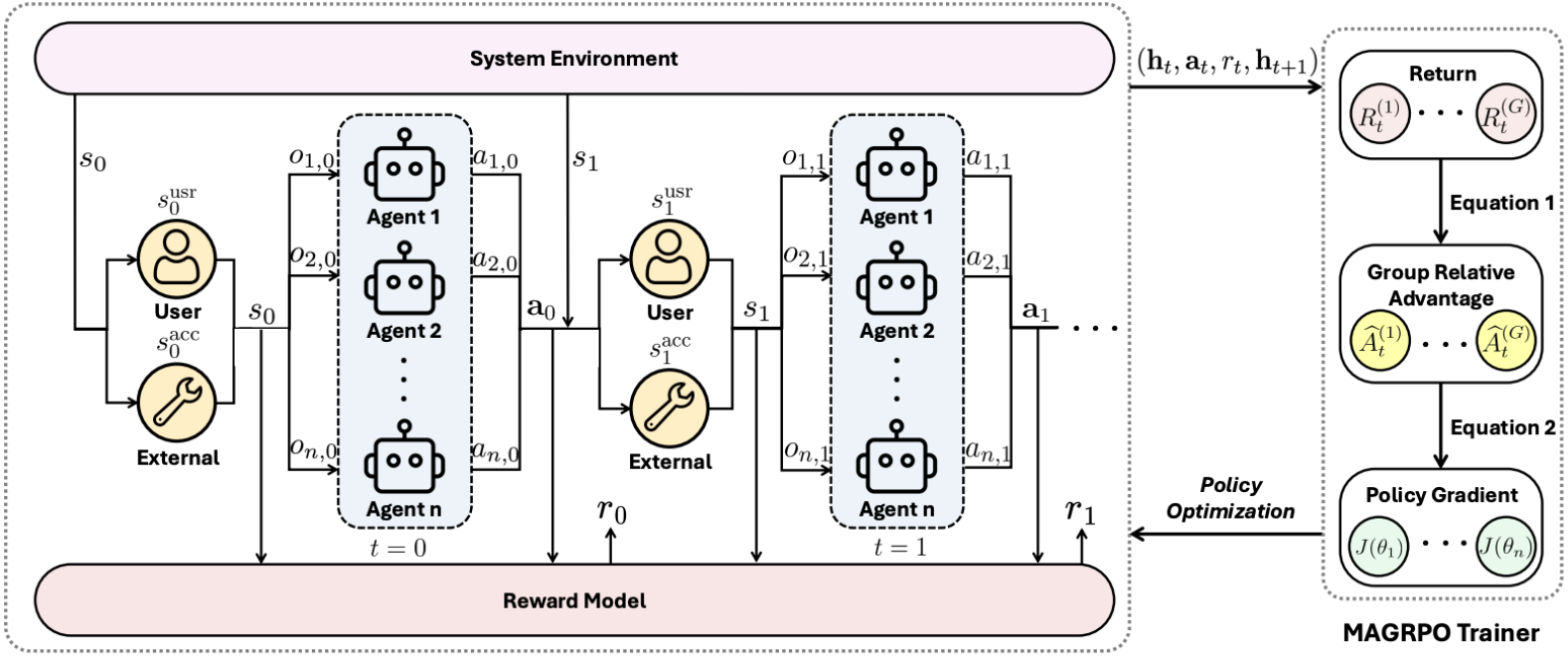
DEBATE, TRAIN, EVOLVE: Self Evolution of Language Model Reasoning (Srivastava et al., 2025)

RL on Multi-Agent Scaffolds



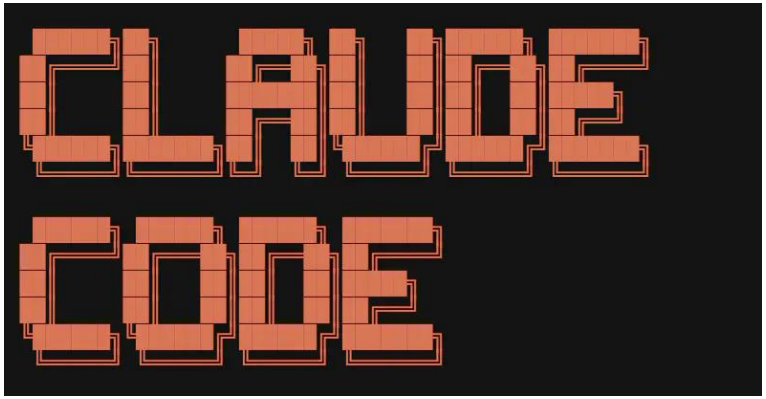
MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning (Park et al., 2025)

RL on Multi-Agent Scaffolds



LLM Collaboration With Multi-Agent Reinforcement Learning (Liu et al., 2025)

Multi-Agent Orchestrators



Open Research Questions

- Verification for Open-ended Problems: Current reinforcement learning methods are mostly designed for verifiable tasks (e.g., math, coding). Can we leverage the models / agents for verification, and what are the *verification scaling laws*?

- Agent Scaffold Architecture: Current agentic scaffolds are mostly designed by humans. Can the agentic scaffold itself be constructed by the model using the task description / environment as input, and the compute allocated accordingly?

- Credit Assignment for Multi-Agent RL: Existing multi-agent RL methods assign uniform credits across agents that contribute to the same scaffold output. How do we assign fine-grained credit to different agents during multi-agent RL?