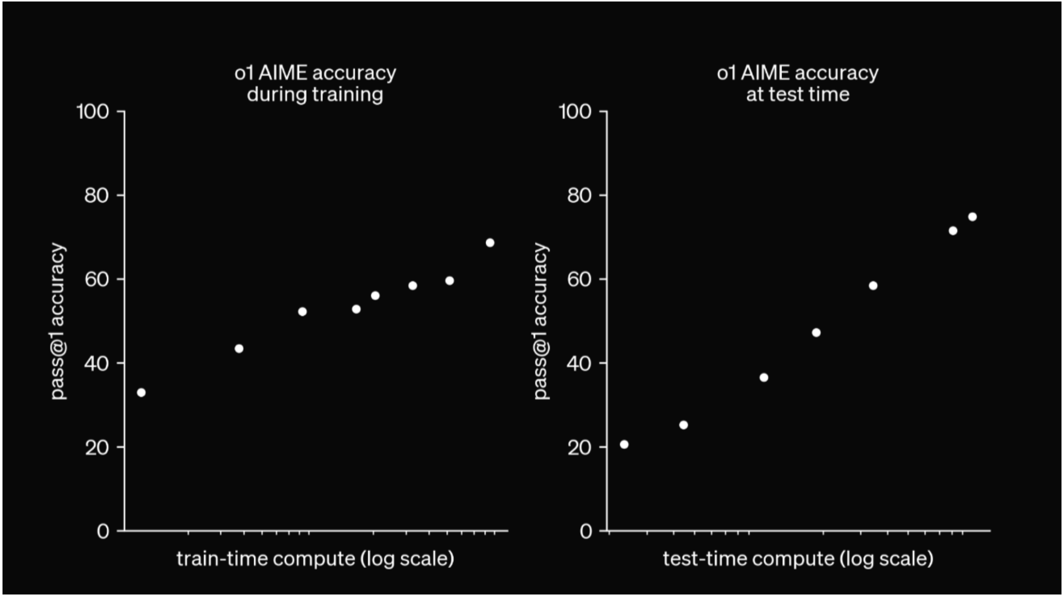
Speculations on Test-Time Scaling

Sasha Rush Daniel Ritter

Cornell

Outline


o1 AIME accuracy
during training
100
80
60
40
20
0
pass@1 accuracy
train-time compute (log scale)
o1 AIME accuracy
at test time
100
80
60
40
20
0
pass@1 accuracy
test-time compute (log scale)

Context

- LLM (2018-2024) driven by training scaling
- Speculation: Benefit of static data running out

Implication

- Breakthrough in large-scale RL Training
-

What have we seen?

- Public demo model
- Strong result in constrained domains.

This Talk

- Survey of the public literature
- Synthesis of discussions with expert
- Gossip and hearsay

Thanks

Lewis Tunstall, Edward Beeching, Aviral Kumar, Charlie Snell, Michael Hassid, Yoav Artzi, Risab Agarwal, Kanishk Gandhi, Wenting Zhao, Yuntian Deng, Nathan Lambert

What we know

Our large-scale **reinforcement learning algorithm** teaches the model how to think productively using its **chain of thought** in a highly **data-efficient** training process.

What we know

- RL - Signal from verifiable problems
- CoT - “Thinking” occurs in token stream
- Data Efficient - Fixed set of good problems

From Gossip

- Single final model
- Not learned from expert examples
-

Chain of Thought

o1 learns to hone its chain of thought and refine the strategies it uses. It learns to recognize and **correct its mistakes**. It learns to **break down tricky steps** into simpler ones. It learns to try a **different approach** when the current one isn’t working.

Review: Chain of Thought

-

Planning

Backtracking

Strategies

Summary

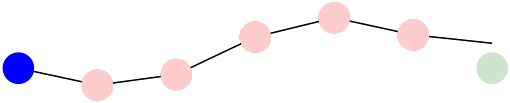
- Solves problems by very long CoT
- CoT includes “thinking” (search / planning)
- Core novelty: Inducing this behavior

Notation - Test-Time (No learning yet!)

- x - the problem specification
- $z \in \mathcal{S}^T$ - the chain of thought (CoT)
- $y \in \mathcal{Y}$ - the final answer
- $p(y|x) = \mathbb{E}_{z \sim p(z|x)} p(y|x, z)$ - model

Warm-up: Ancestral Sampling

- $\tilde{z} \sim p(z|x)$
- $\tilde{y} \sim p(y|x, z = \tilde{z})$

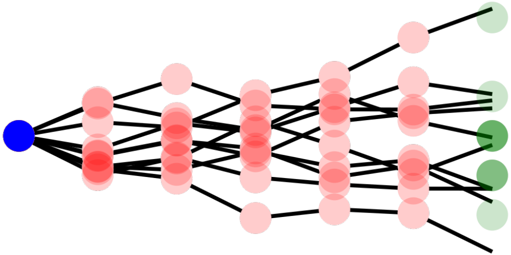


$|\tilde{z}|$ is the amount of test-time compute

Warm-up: Monte-Carlo (Self-Consistency)

- $\tilde{z} \sim p(z|x)$
- $\tilde{y} \sim p(y|x, \tilde{z})$

Pick majority choice $\tilde{y}^i$

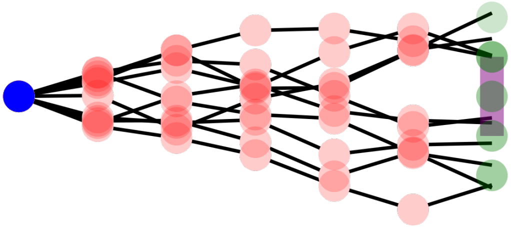


Notation - Verifier

- $\text{Ver} : \mathcal{Y} \to \{0, 1\}$, tells us if an answer is correct or not
- Examples: Regular expression for math, unit test for code.

Warm up: Rejection Sampling / Best-of-N

- $\tilde{z} \sim p(z|x)$
- $\tilde{y} \sim p(y|x, \tilde{z})$



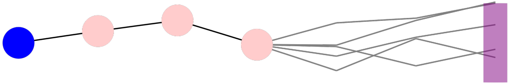
We only keep the correct subset of $\tilde{y}$, $\{\tilde{y} : Ver(\tilde{y})\}$

Variants:

Warm up: Monte-Carlo Roll-Outs

- Given $x, z_{1:t}$ (a partial chain of thought) define expected reward as

$$\mathbb{E}_{y \sim p(y|z,x), z_{t:T} \sim p(z|z_{1:t},x)}[Ver(y)]$$



- Roll-outs apply MC to this expectation.

Goal: Learning

- $\max_\theta \sum \log p(y|x;\theta)$

- Intractable expectation over latent CoT

Outline

Introduction

The Clues

Notation

The Suspects

No Verifier

Conclusions

The Suspects

- Guess + Check
- Guided Search
- AlphaZero
- Learn to Search
- Wildcard

Informal: Guess + Check

- Sample N CoTs
- Check if successful
- Train on good ones

Formalization: Rejection Sampling EM

$$\max_{\theta} \sum \log p(y|x;\theta) = \sum \log E_z p(y,z|x)$$

- E-Step: Sample $\mathcal{Z} = \{\tilde{z}_i\}_{i=1}^{N}$ from the posterior with Rejection Sampling

$$\tilde{z} \sim p(z|\textsf{Ver}(y) = 1, x)$$

- M-Step: Fit $\theta' \leftarrow \arg\max_\theta \sum_{\tilde{z}\in\mathcal{Z}} \log p(\tilde{z}|x;\theta)$

Terminology

[Zelikman et al., 2022, Gulcehre et al., 2023, Singh et al., 2023]

- STaR
- ReST
- ReST-EM
- Filtered Rejection Sampling
- Best-of-N Training

Batched

- Batched -> Compute trajectories first, then train with behavioral cloning

- Online -> Use policy gradient-like steps to update after each example

Empirical Results

Find a good chart from a paper. Best representative.
Improvement over self-consistency.

Why might this be right?

- Extremely simple and scalable

- Good baseline in past work

- No evidence this learns to correct, plan

- Well-explored in literature with marginal gains

Deeper

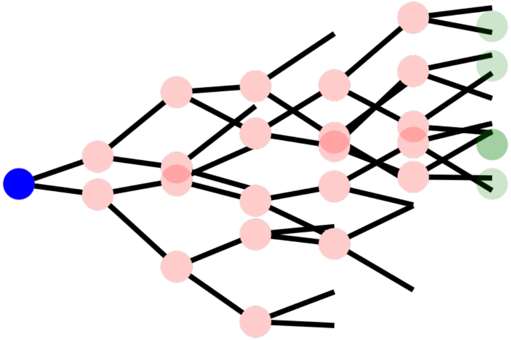
- Rejection sampling may be really inefficient.
- Particularly on hard problems, may get no signal

Informal: Guided Search

- During CoT sampling, use a “guide” to correct trajectories
- Check if final versions are successful
- Train on good ones

Beam Search with Guide

- $r : S^T \to \mathbb{R}$ is the guide/reward function
- Take k samples $\{\tilde{z}_t^i\}_{i=1}^k$ from $p(z_t | x, z_{1:t-1})$
- Keep the top m samples, ordered by $r(\tilde{z}_t^i)$
- Repeat for each $\tilde{z}_t^i$ until generation ends

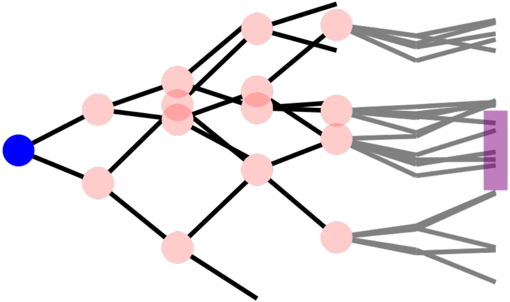


What to use as Guide?

- Monte Carlo Roll-outs
- Learned Reward Model

Beam Search with Roll-Outs

- For one $\tilde{z}_t^i$, sample n solutions $\tilde{y}^{i,j}$ from $p(y, z_{t+1:T} | x, z_{1:t-1}, \tilde{z}_t^i)$
- $r_{MC}(\tilde{z}_t) = \frac{1}{n} \sum_{j=1}^{n} Ver(\tilde{y}^{i,j})$

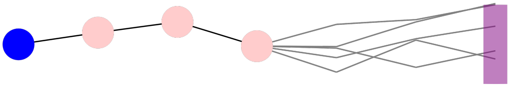


Amortized Roll-Outs

[Wang et al., 2023]

- Rollouts are costly, so instead learn a model $r_\psi(z_t)$ to approximate rollouts

- Use r_{MC} to generate labels to train r_ψ



What about test time?

- Learned rewards can improve test-time without verifier.
-

Terminology

[Uesato et al., 2022, Setlur et al., 2024, Wang et al., 2023, Lightman et al., 2023, Snell et al., 2024]

- Value
- PRM
- PAV
- Math Shepard.
- snell.

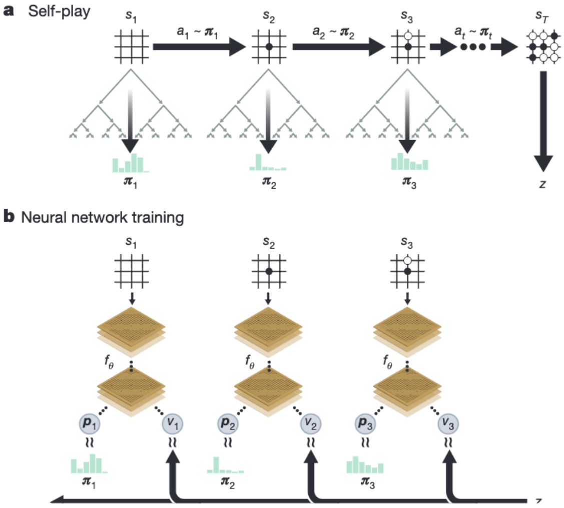
Why might this be right?

- OpenAI is exploring
- Makes RS more efficient.
- Learned rewards are effective
- Assumption: o1 is a single test-time model (although could train or distill-in)
- Not clear if it learns planning.

Deeper

- Improving search seems critical.

Reminder: AlphaZero



Informal: AlphaZero

- Self-play using guided-search with exploration
- Label final outcomes of self-play games
- Train both a reward model and policy

[Anthony et al., 2017]

Formalized: Expert Iteration

- Iterative algorithm combining learned model + expert search with a verifier.

- Generate samples using $p(y, z|x)$, reward model $r(z_t)$, and search algorithm (e.g. beam search)

- Label samples using $Ver(y)$

- Train $p(y, z|x)$, $r(z_t)$ on the labeled samples

UCB for exploration

-

Empirical Results

Find a good chart from a paper. Best representative. Improvement over STarish.

Why might this be right?

- Major demonstrated RL result
-
-
-

Deeper

- Can we force the model to search?

Informal: Learning to Correct

- Sample N Successful CoTs
- Edit $z \to z'$ to inject incorrect expansions before correct ones.
- Train on z' trajectories

Formalized: Stream of Search

[Gandhi et al., 2024]

-
-

Empirical Results

Score results

Why might this be right?

-
-

Why might this be wrong?

-
-

[Brown et al., 2024]

No verifier

•

Reference I

[Anthony et al., 2017] Anthony, T., Tian, Z., and Barber, D. (2017).
Thinking fast and slow with deep learning and tree search.
arXiv [cs.AI].

[Brown et al., 2024] Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. (2024).
Large language monkeys: Scaling inference compute with repeated sampling.
arXiv [cs.LG].

Reference II

[Gandhi et al., 2024] Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. (2024).
Stream of search (SoS): Learning to search in language.
arXiv [cs.LG].

[Gulcehre et al., 2023] Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., Macherey, W., Doucet, A., Firat, O., and de Freitas, N. (2023).
Reinforced self-training (ReST) for language modeling.
arXiv [cs.CL].

Reference III

[Lightman et al., 2023] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023).
Let's verify step by step.
arXiv [cs.LG].

[Setlur et al., 2024] Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. (2024).
Rewarding progress: Scaling automated process verifiers for LLM reasoning.
arXiv [cs.LG].

Reference IV

[Singh et al., 2023] Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., Kumar, A., Alemi, A., Rizkowsky, A., Nova, A., Adlam, B., Bohnet, B., Elsayed, G., Sedghi, H., Mordatch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J., Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp, L., Xiao, L., Bileschi, M. L., Constant, N., Novak, R., Liu, R., Warkentin, T., Qian, Y., Bansal, Y., Dyer, E., Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N. (2023).
Beyond human data: Scaling self-training for problem-solving with language models.

Reference V

arXiv [cs.LG].

[Snell et al., 2024] Snell, C., Lee, J., Xu, K., and Kumar, A. (2024).
Scaling LLM test-time compute optimally can be more effective than scaling model parameters.
arXiv [cs.LG].

[Uesato et al., 2022] Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. (2022).
Solving math word problems with process- and outcome-based feedback.

Reference VI

arXiv [cs.LG].

[Wang et al., 2023] Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. (2023).
Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations.
arXiv [cs.AI].

[Zelikman et al., 2022] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. (2022).
STaR: Bootstrapping reasoning with reasoning.
arXiv [cs.LG].