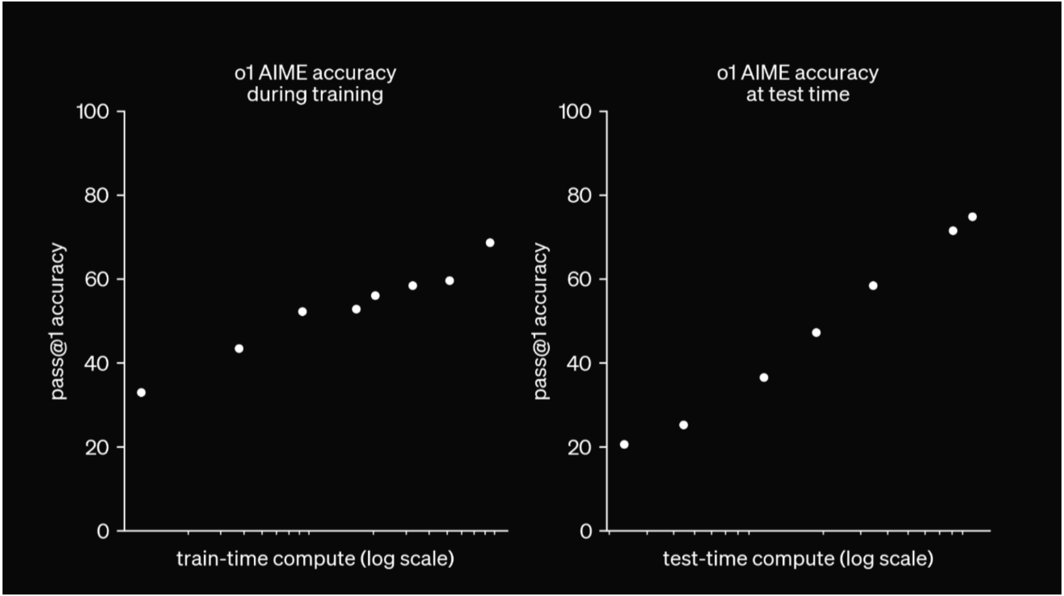
Speculations on Test-Time Scaling

Sasha Rush Daniel Ritter

Cornell

Outline


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

First, the cipher seems connected to the plaintext. Given the time constraints, perhaps the easiest way is to try to see patterns. Option 1: Try to find mapping from letters to letters. Do any letters match? First, let's write down the ciphertext and plaintext letters on top of each other.

Backtracking

Strategies

Summary

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

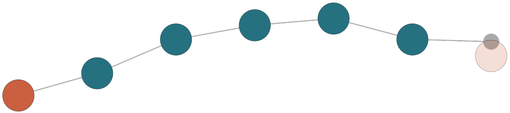
Notation: LLM Sampling (No learning yet!)

- x; problem specification
- $z_{1:T} \in \mathcal{S}^T$; chain of thought (CoT) stepsl
- $y \in \mathcal{Y}$; final answer

$$p(y|x) = \mathbb{E}_z p(y|x, z)$$

Warm-up: Ancestral Sampling

$$z_{1:T} \sim p(\cdot | x)$$
$$y \sim p(\cdot | x, z_{1:T})$$



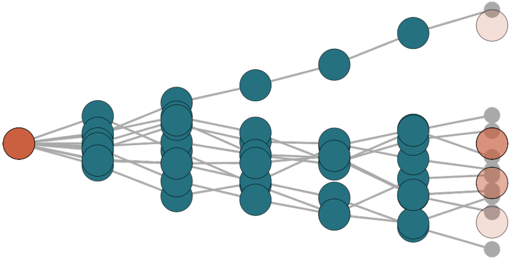
T is the amount of test-time compute

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

For N samples,

$$z_{1:T} \sim p(\cdot|x)$$
$$y^n \sim p(\cdot|x, z_{1:T})$$

Pick majority choice y^n



Assumption: Verifier

$$\text{Ver}_x : \mathcal{Y} \to \{0, 1\}$$

Examples:

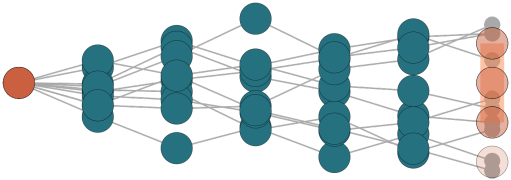
- Regular expression for math
- Unit test for code
- Test questions for science

Warm up: Rejection Sampling / Best-of-N

For $n = 1$ to N:

$$z^n \sim p(z|x)$$
$$y^n \sim p(y|x, z^n)$$

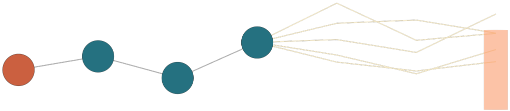
Verified set $\{y^n : \text{Ver}(y^n)\}$



Warm up: Monte-Carlo Roll-Outs

Given partial CoT $z_{1:t}$, expected value,

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



Rollout = Monte Carlo for this expectation.

Goal: Learning with Latent CoTs

Maximum likelihood;

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

Classic combinatorial expectation

Outline

Introduction

The Clues

Notation

The Suspects

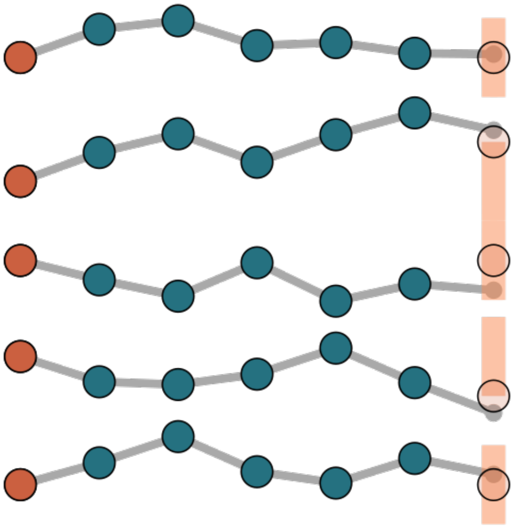
Conclusions

The Suspects

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

Suspect 1: Guess + Check

- 1) Sample N CoTs
- 2) Check if successful
- 3) Train on good ones



G+C Formalization: Rejection Sampling EM

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

- E-Step: For $n = 1$ to N:

$$z^n \sim p(\cdot|x)$$
$$y^n \sim p(\cdot|x, z^n)$$

  Keep verified set $\mathcal{Z} = \{z^n : \text{Ver}(y^n)\}$

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

G+C Variants

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

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

G+C Variants

- Batched -> Compute trajectories first, then train with behavioral cloning
- Online -> Use policy gradient-like steps to update after each example

G+C Empirical Results

[Singh et al., 2023]

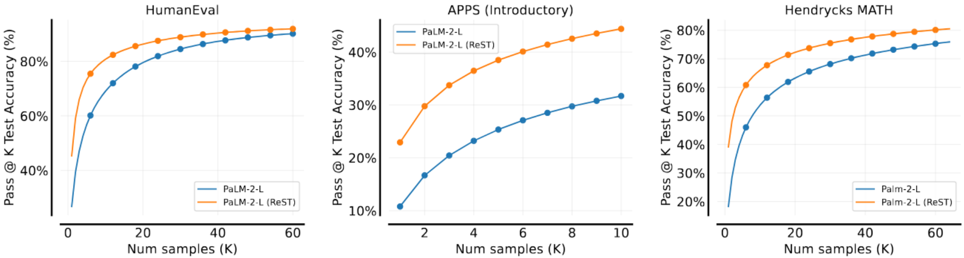


Figure 5 | **Pass@K results** for PaLM-2-L pretrained model as well as model fine-tuned with ReST^{EM}. For a fixed number of samples K, fine-tuning with ReST^{EM} substantially improves Pass@K performance. We set temperature to 1.0 and use nucleus sampling with $p = 0.95$.

G+C 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

More Structure?

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

Suspect 2: Guided Search

- During CoT sampling, use a heuristic to improve trajectories

- Check if final versions are successful

- Train on good ones

GS: Beam Search with Guide

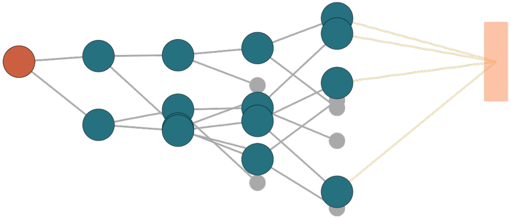
$r : \mathcal{S}^t \to \mathbb{R}$; Guide function

For each step t

1. Sample next step,

$$z_t \sim p(\cdot | x, z^i_{1:t-1})$$

2. Keep the top N samples, ordered by $r(z_t)$



What to use as Guide?

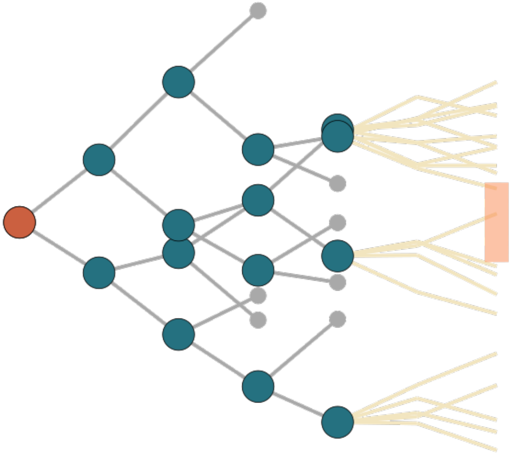
- Monte Carlo Roll-outs
- Learned Value Function

Beam Search with Roll-Outs

For a z_t, sample answers

$$y^n \sim p(\cdot | x, z_{1:t-1})$$

$$r_{MC}(z_t) = \frac{1}{N} \sum_{i=1}^{n} \text{Ver}(y^i)$$

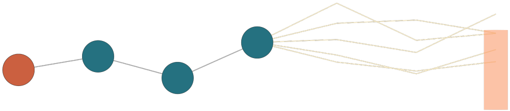


Amortized Roll-Outs

[Wang et al., 2023]

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

- Use r_{MC} to determine 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.

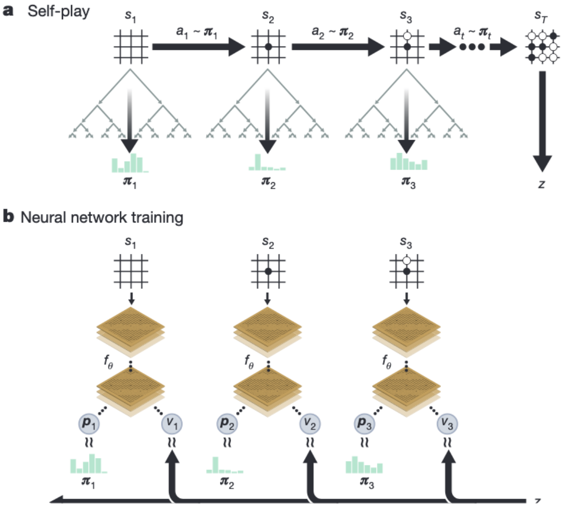
Why might this be right?

- 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.

More Structure

- Improving search seems critical.

Reminder: AlphaZero



Suspect 3: AlphaZero

- Self-play using guided-search with exploration
- Label final outcomes of self-play games
- Train guide and generator

[Anthony et al., 2017]

Formalized: Expert Iteration

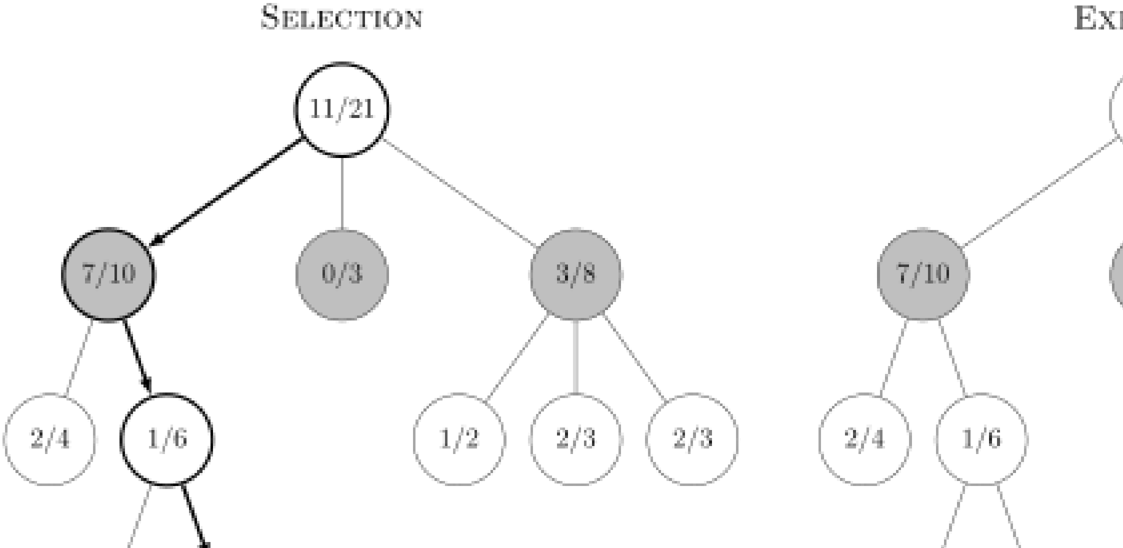
- Iterative algorithm combining learned model + expert search with a verifier.

- 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, and repeat

MCTS exploration



UCB for Language

- Selection: Walk down tree to leaf z_{t-1}
- Expand: Sample K next steps z_t^i, pick one at random
- Rollouts: Sample $z_{t+1} \dots z_T$
- Backprop: Update nodes counts $z_{1:t}$ based on results

Compared with Search

- System builds in exploration
- Scales to more train-time search

- Costly to maintain open states
- More complex algorithmically

Empirical Results

[?]

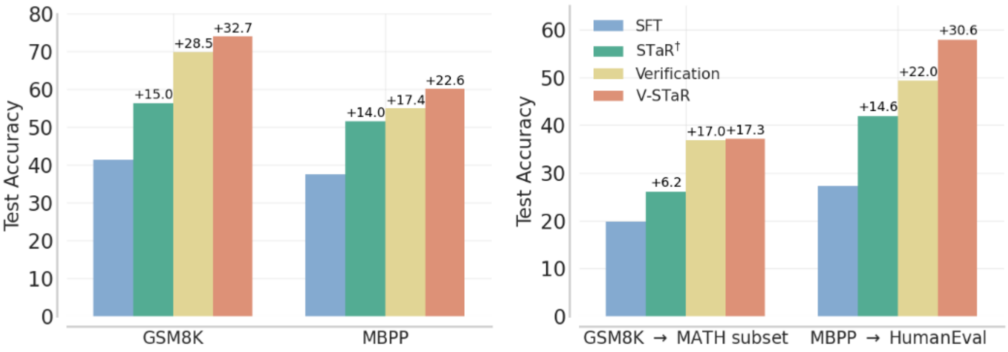


Figure 8: Test accuracy of 13B V-STaR compared to baselines. We report Best-of-64 for verification-based methods and Pass@1 for others. **(Left)** Test accuracy for training tasks. **(Right)** Transfer evaluation of GSM8K and MBPP trained models on MATH subset and HumanEval respectively.

Why might this be right?

- Major demonstrated RL result
-
-
-

More Structure

- Can we force the model to search?

Suspect 4: Learning to Correct

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

[Gandhi et al., 2024]

Formalized: Stream of Search

-
-

Empirical Results

Score results

Why might this be right?

-
-

Why might this be wrong?

-
-

Less Structure?

- Maybe this is all too much...

[Brown et al., 2024]

Suspect 5: Compute Injection

-

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]*.