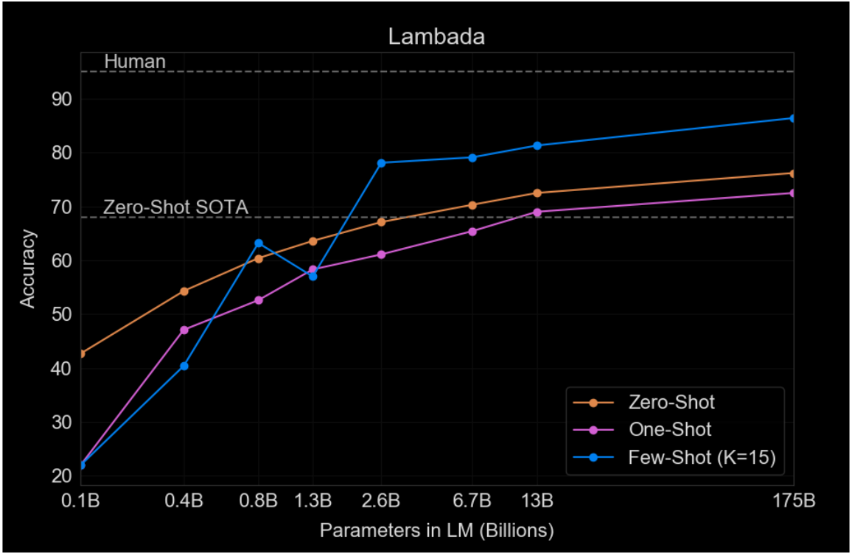
Speculations on Test-Time Scaling

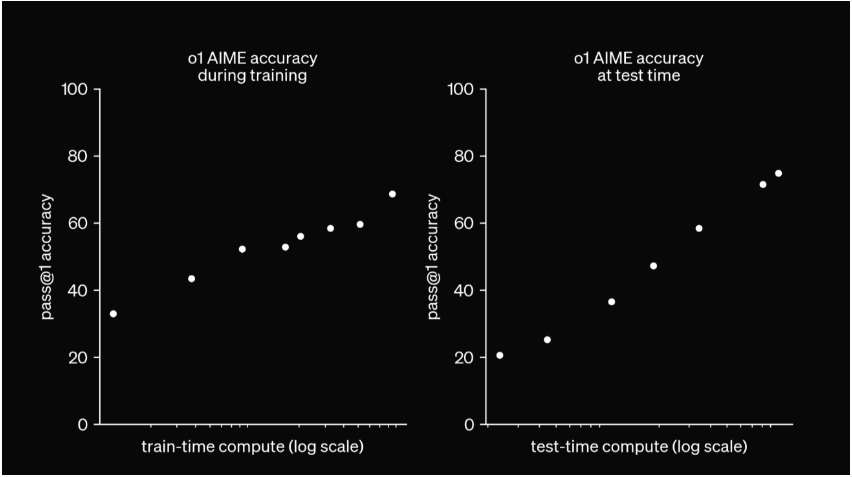
Sasha Rush Daniel Ritter

Cornell - Hugging Face

[Brown et al., 2020]



[OpenAI, 2024]


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)

AIME

[Hendrycks et al., 2021c]

For any finite set X, let $|X|$ denote the number of elements in X. Define

$$S_n = \sum |A \cap B|,$$

where the sum is taken over all ordered pairs (A, B) such that A and B are subsets of $\{1, 2, 3, \cdots, n\}$ with $|A| = |B|$. For example, $S_2 = 4$ because the sum is taken over the pairs of subsets

$$(A, B) \in \{(\emptyset, \emptyset), (\{1\}, \{1\}), (\{1\}, \{2\}), (\{2\}, \{1\}), (\{2\}, \{2\}), (\{1, 2\}, \{1, 2\})\}$$

giving $S_2 = 0 + 1 + 0 + 0 + 1 + 2 = 4$. Let $\frac{S_{2022}}{S_{2021}} = \frac{p}{q}$, where p and q are relatively prime positive integers. Find the remainder when $p + q$ is divided by 1000.

[Sutton, 2019]

The Bitter Lesson

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**.

Importance of Search

[Brown and Sandholm, 2017], https://x.com/polynoamial/status/1840822629625688469

The most important [lesson] is that I and other researchers simply didn't know how much of a difference scaling up search would make. If I had seen those scaling results at the start of my PhD, I would have shifted to researching search algorithms for poker much sooner and we probably would have gotten superhuman poker bots much sooner.

Scaling Laws for Board Games

[Jones, 2021]

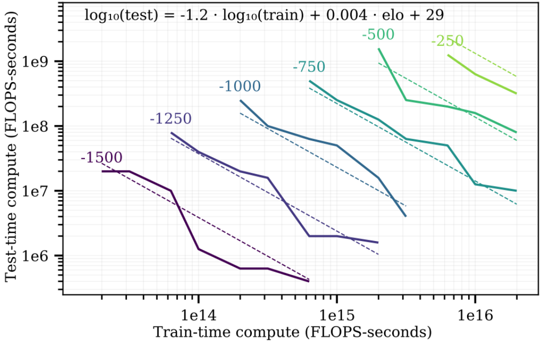


Fig. 9. The trade-off between train-time compute and test-time compute. Each dotted line gives the minimum train-test compute required for a certain Elo on a 9×9 board

Search Against Learned Verifiers

[Cobbe et al., 2021]

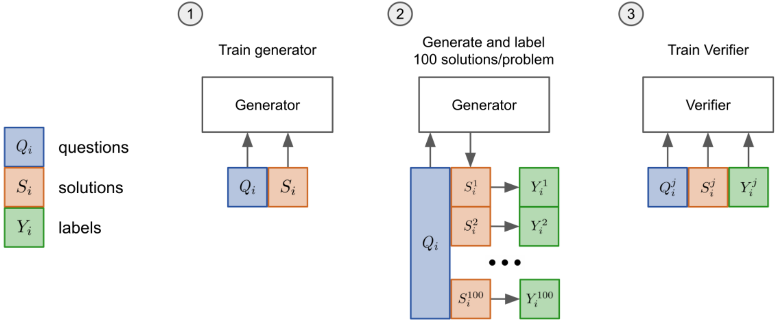
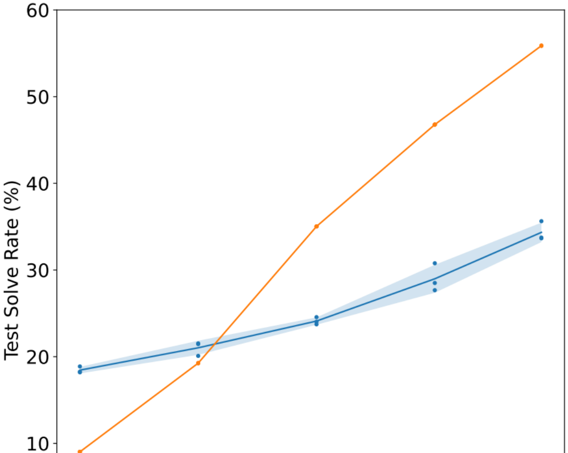


Figure 4: A diagram of the verification training pipeline.

Search Against Learned Verifiers

[Cobbe et al., 2021]



Sources

- Synthesis of discussions with expert
- Survey of the public literature
- Rumors from social media

Thanks to Lewis Tunstall, Edward Beeching, Aviral Kumar, Charlie Snell, Michael Hassid, Yoav Artzi, Niklas Muennighoff, Risab Agarwal, Rafael Rafailov, Kanishk Gandhi, Wenting Zhao, Yuntian Deng, Nathan Lambert, Noah Goodman, Thomas Ahle, Li Erran Li, Justin Chiu, Jacob Eisenstein, Sean Welleck, Caglar Gulcehre, Ofir Press

Outline

Introduction

The Clues

Technical Background

The Suspects

Implications

[OpenAI, 2024]

o1 Description

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.

Implication

- RL; Signal from verifiable problems
- CoT; Test-time occurs in token stream
- Data-Efficient; Bounded set of problems

Current Assumptions

- Single final language model with coherent CoT
- Not following from expert examples
- Behaviors are learned

Review: Chain of Thought

[Wei et al., 2022, Nye et al., 2021]

- The model generates intermediate steps.
- These steps provide an scratchpad for technical problems.

Question: 4 baskets. 3 have 9 apples, 15 oranges, 14 bananas each. 4th has 2 less of each. Total fruits?
Let's solve step-by-step:
Fruits in one of first 3 baskets: 9 + 15 + 14 = 38
Total in first 3 baskets: 38 * 3 = 114
4th basket: (9-2) + (15-2) + (14-2) = 32
Total fruits: 114 + 32 = 146

Answer: 146 fruits

[OpenAI, 2024]

o1 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.

o1 CoT: Outlining

[OpenAI, 2024]

Implementation Outline:
1. Capture input string as argument.
2. Remove any spaces (if any).
3. Parse input string to extract numbers as arrays.
- Since the input is in the format '[1,2],[3,4]', we can:
- Remove outer brackets if necessary.
- Split the string by '],' to get each row.
- For each row, remove '[' and ']', then split by ',' to get elements.
4. Build a 2D array in bash (arrays containing arrays).

[OpenAI, 2024]

o1 CoT: 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.

o1 CoT: Backtracking

[OpenAI, 2024]

Similarly, m(x) (−x 2) = (−x2n + 2 + m2n−2x2n + lower terms)m(x) * (−x 2) = (−x 2n + 2 + m 2n−2 x 2n + lower terms).*

Wait, actually, this may not help us directly without specific terms.An alternative is to consider the known polynomials.

So m(x) = k ...

o1 CoT: Self-Evaluation

[OpenAI, 2024]

Let's analyze each option.

Option A: "because appetite regulation is a field of staggering complexity."

Is that a good explanation? Hmm.

Option B: "because researchers seldom ask the right questions."

Does this make sense with the main clause?

Summary

- CoT provides test-time scaling
- CoT looks like search / planning in a classical sense
- RL needed to induce this behavior

Outline

Technical Background

[Cobbe et al., 2021]

- Formalize sampling of latent reasoning
- No learning yet.

Question: 4 baskets. 3 have 9 apples, 15 oranges, 14 bananas each. 4th has 2 less of each. Total fruits?

1 - Let's solve step-by-step:

2 - Fruits in one of first 3 baskets: 9 + 15 + 14 = 38

3 - Total in first 3 baskets: 38 * 3 = 114

4 - 4th basket: (9-2) + (15-2) + (14-2) = 32

5 - Total fruits: 114 + 32 = 146

Answer: 146 fruits

Stepwise CoT Sampling

[Welleck et al., 2024]

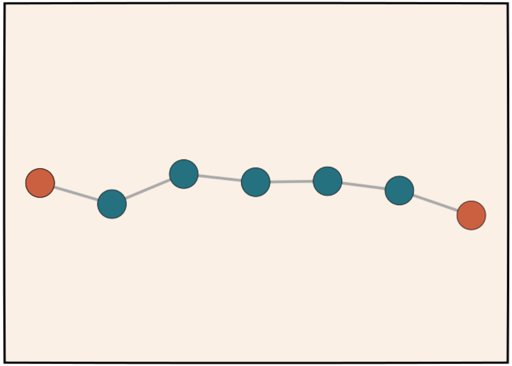
- x; problem specification
- $z_{1:T} \in \mathcal{S}^T$; chain of thought (CoT) steps
- $y \in \mathcal{Y}$; final answer

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

Warm-up: Ancestral Sampling

[Wei et al., 2022]

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



T is the amount of test-time compute

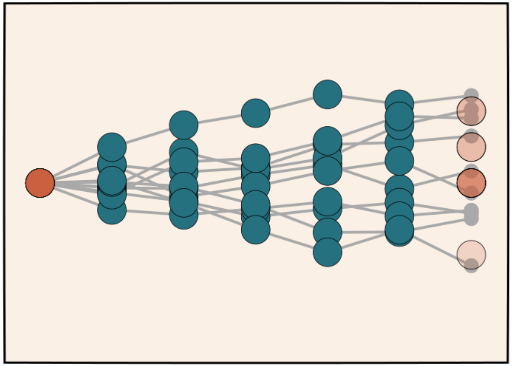
[Wang et al., 2022]

Self-Consistency / Majority Vote

For N samples,

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

Pick majority choice y^n



Assumption: Automatic Verifier at Training

[Uesato et al., 2022]

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

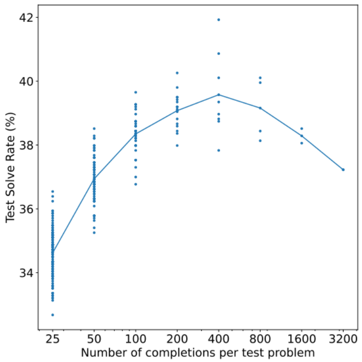
Common datasets:

- Regular expression for math [Cobbe et al., 2021]
- Unit test for code [Hendrycks et al., 2021a]
- Test questions for science [Hendrycks et al., 2021b]

[Cobbe et al., 2021, Lightman et al., 2023]

Automatic Verifier?

- OpenAI primarily interested in learned verifiers (ORM)
- Spec: Large-scale annotation of on-policy outputs



(a) 6B verification test performance when given varying numbers of completions per problem to rank.

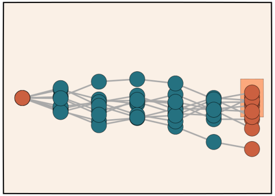
[Nakano et al., 2021, Lightman et al., 2023]

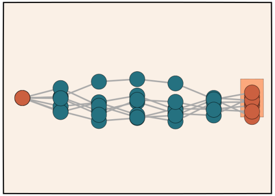
Rejection Sampling
Best-of-N

For $n = 1$ to N:

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

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





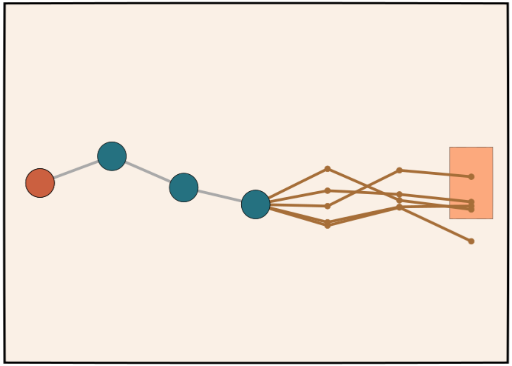
Monte-Carlo Roll-Outs

[Wang et al., 2023]

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

$$\mathbb{E}_{y \sim p(\cdot|x)} \mathrm{Ver}(y)$$

Monte Carlo for this expectation.



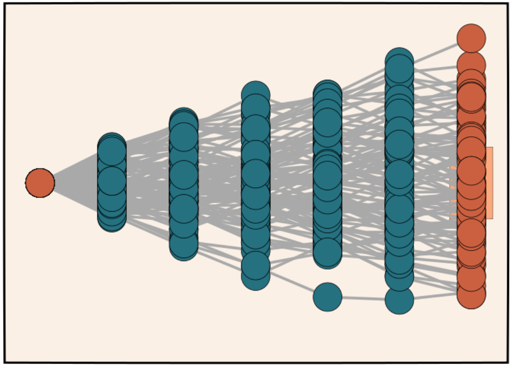
Goal: Learning with Latent CoTs

Maximum likelihood;

$$\max_{\theta} \sum \log p(\mathsf{Ver}(y)|x;\theta) =$$

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

Classic combinatorial expectation



Reinforcement Learning

Important practical choices:

- Batched? $\rightarrow$ Compute trajectories first, then train
- On-policy? $\rightarrow$ Sample from current model
- KL Constraints on learning.
- Specific algorithm choice (REINFORCE, PPO, etc)

https://www.youtube.com/watch?v=tEzs3VHyBDM

When training a model for reasoning, one thing that immediately jumps to mind is to have humans write out their thought process and train on that. When we saw that if you train the model using RL to generate and hone its own chain of thoughts it can do even better than having humans write chains of thought for it. That was the “Aha!” moment that you could really scale this.

Outline

Introduction

The Clues

Technical Background

The Suspects

Implications

The Suspects

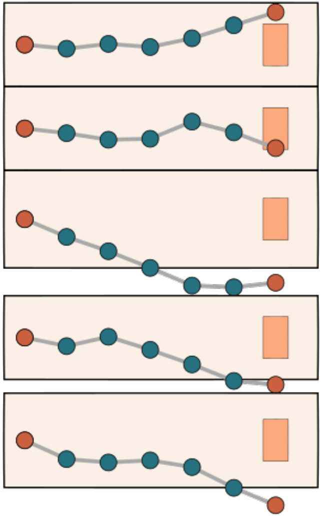
- Guess + Check
- Process Rewards
- Search / AlphaZero
- Learning to Correct

The Suspects

- Guess + Check
- Process Rewards
- Search / AlphaZero
- Learning to Correct

Suspect 1: Guess + Check

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



Framework: Rejection Sampling EM

[Neal and Hinton, 1998]

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

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

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

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

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

Variants

- Self-Training [Yarowsky, 1995]
- Best-of-N Training [Cobbe et al., 2021]
- STaR [Zelikman et al., 2022]
- ReST [Gulcehre et al., 2023]
- ReST-EM [Singh et al., 2023]
- Filtered Rejection Sampling [Nakano et al., 2021]

Empirical Results

[Singh et al., 2023]

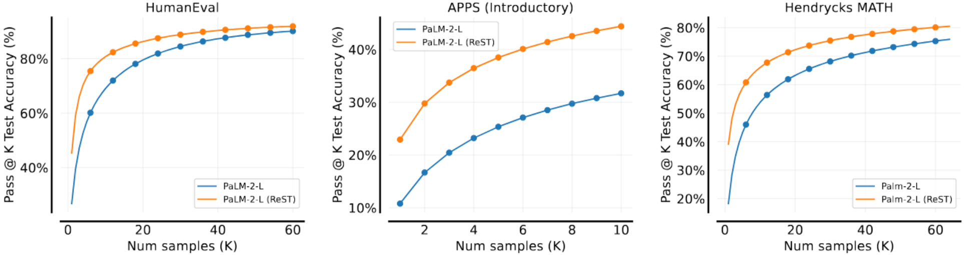


Figure 5 | **Pass@K results** for PaLM-2-L pretrained model as well as model fine-tuned with ReST^{EM}. For a fixed number of samples K, fine-tuning with ReST^{EM} substantially improves Pass@K performance. We set temperature to 1.0 and use nucleus sampling with $p = 0.95$.

Learned Verifier

- Ver available only at train

- Samples can be used to further train a learned verifier (amortization)

- Can be used for test-time rejection sampling.

Is this o1?

Pro

- ✓ Extremely simple and scalable
- ✓ Positive results in past work

Is this o1?

Pro

- ✓ Extremely simple and scalable
- ✓ Positive results in past work

- × No evidence this learns to correct, plan
- × Computationally inefficient search

The Suspects

- Guess + Check
- Process Rewards
- AlphaZero
- Learning to Correct

The Suspects

- Guess + Check
- Process Rewards
- AlphaZero
- Learning to Correct

Suspect 2: Process Rewards

- 1) During CoT sampling, use guidance to improve trajectories
- 2) Check if final versions are successful
- 3) Train on good ones

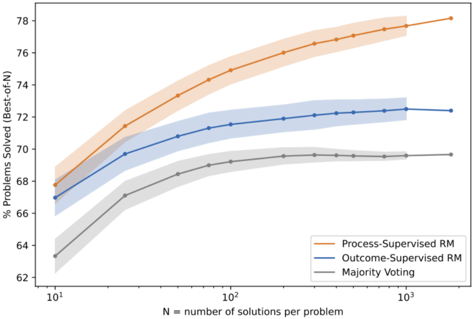
Process Rewards

[Uesato et al., 2022, Lightman et al., 2023]

- Early learned verification (PRM) improves over learned verification (ORM)

$$r : S^t \to \mathbb{R}$$

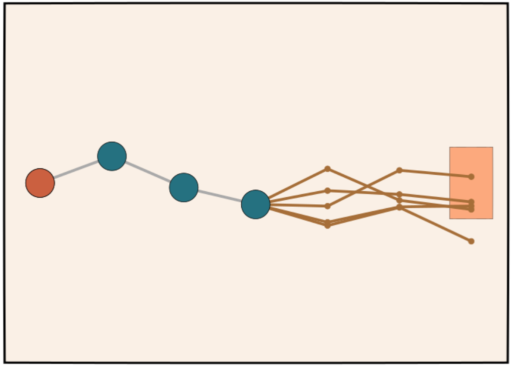
	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6



Learned Process Rewards

[Lightman et al., 2023, Wang et al., 2023]

- Rollouts are costly / require Ver
- Learn $r_\psi(z_t)$ to approximate
- Use Monte-Carlo for labels



[Zhang et al., 2024, Ankner et al., 2024]

Generative Verifiers

- Define $r_\psi(z_t)$ as an LLM
- Merges process reward with generation
- Note: allows for verification CoT

Open Process Rewards

[Wang et al., 2023]

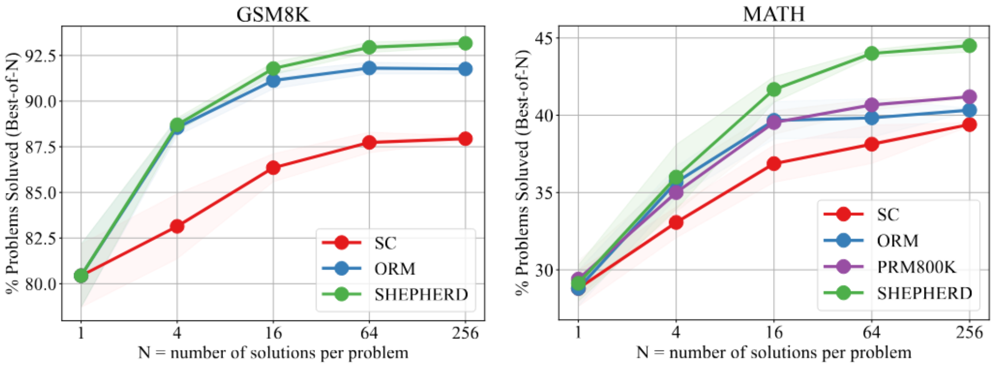


Figure 3: Performance of LLaMA2-70B using different verification strategies across different numbers of solution candidates on GSM8K and MATH.

Incorporating at Test-Time

- Process Reward does not need Ver can be used at test-time.
- If generative, can be merge into a singel CoT stream

Let's analyze each option.

Option A: "because appetite regulation is a field of staggering complexity."

Is that a good explanation? Hmm.

Is this o1?

✓ Intermediate guides are effective

✓ Removes challenges of full learned verifier

Is this o1?

- ✓ Intermediate guides are effective
- ✓ Removes challenges of full learned verifier

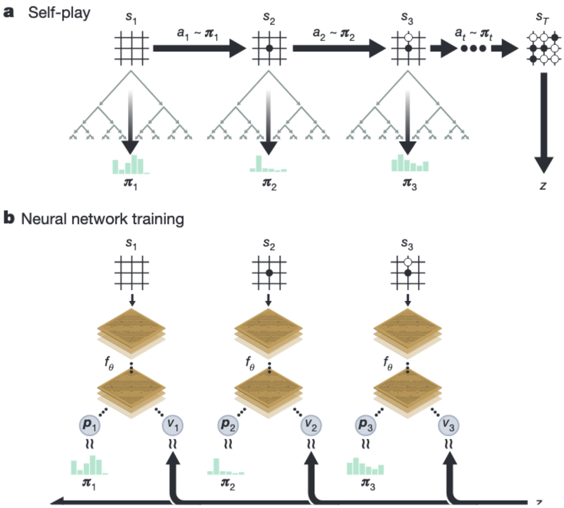
- × Not clear if this is enough for planning.
- × Need to combine generator / guide into one CoT

The Suspects

- Guess + Check
- Process Rewards
- Search / AlphaZero
- Learning to Correct

Reminder: AlphaZero

[Silver et al., 2017]



- Canonical example of self-learning
- Scaling model without data

AlphaProof

[Silver et al., 2017]

When presented with a problem, AlphaProof generates solution candidates and then proves or disproves them by searching over possible proof steps in Lean. Each proof that was found and verified is used to reinforce AlphaProof's language model, enhancing its ability to solve subsequent, more challenging problems.

Suspect 3: AlphaZero

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

[Anthony et al., 2017]

Framework: Expert Iteration

- Iterative algorithm combining learned model + expert search with a verifier.
- Generate samples using $p(y, z|x)$, reward model $g(z_t)$, and search algorithm (e.g. beam search)
- Label samples using $\text{Ver}_x(y)$
- Train $p(y, z|x)$, $r(z_t)$ on the labeled samples, and repeat

Framework: Beam Search with Guide

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

Framework: Beam Search with Guide

[Snell et al., 2024]

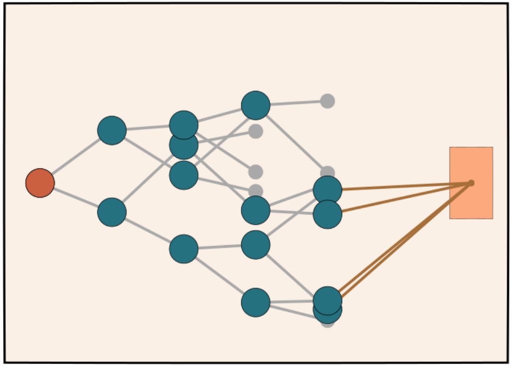
$$r : \mathcal{S}^t \to \mathbb{R}$$

For each step t,

1. Sample many next steps,

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

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



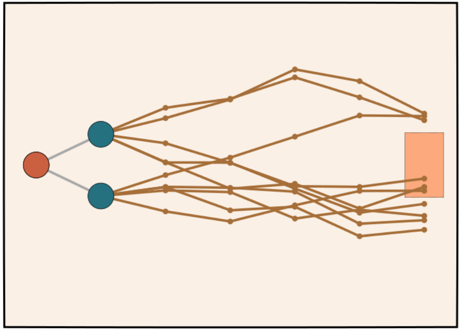
[Kazemnejad et al., 2024]

Beam Search with Roll-Outs

For partial $z_{1:t-1}$, rollout,

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

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



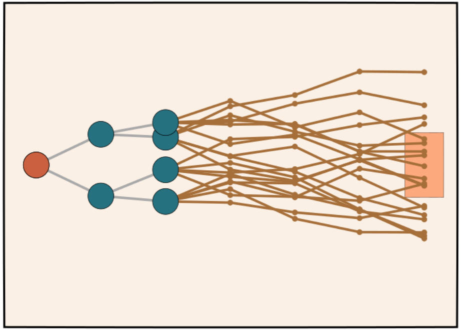
Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

For partial $z_{1:t-1}$, rollout,

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

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

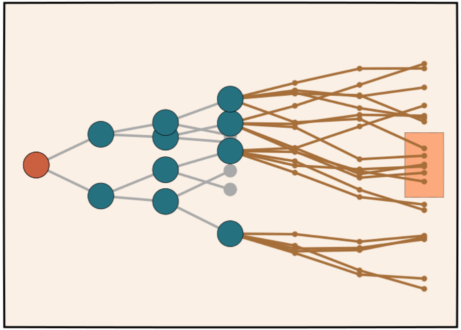


Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

For partial $z_{1:t-1}$, rollout,

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



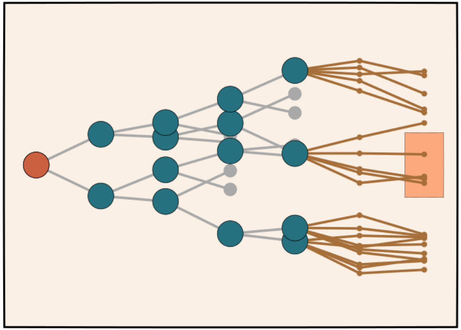
$$r_{MC}(z_t) = \frac{1}{N} \sum_{n=1}^{N} \text{Ver}(y^n)$$

Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

For partial $z_{1:t-1}$, rollout,

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



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

Empirical Results: Expert Iteration

[Hosseini et al., 2024]

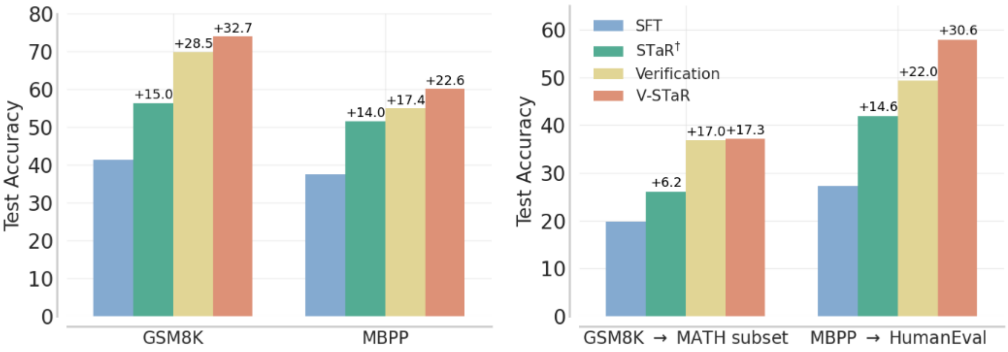


Figure 8: Test accuracy of 13B V-STaR compared to baselines. We report Best-of-64 for verification-based methods and Pass@1 for others. **(Left)** Test accuracy for training tasks. **(Right)** Transfer evaluation of GSM8K and MBPP trained models on MATH subset and HumanEval respectively.

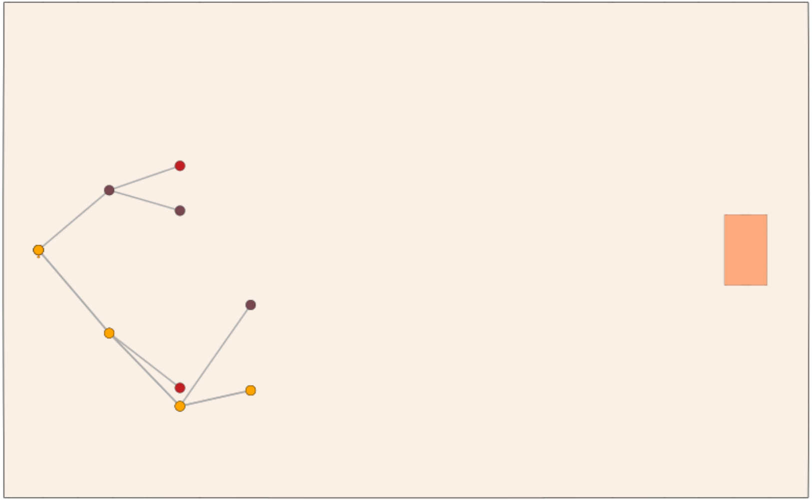
[Hubert et al., 2021, Feng et al., 2023]

MCTS for Language

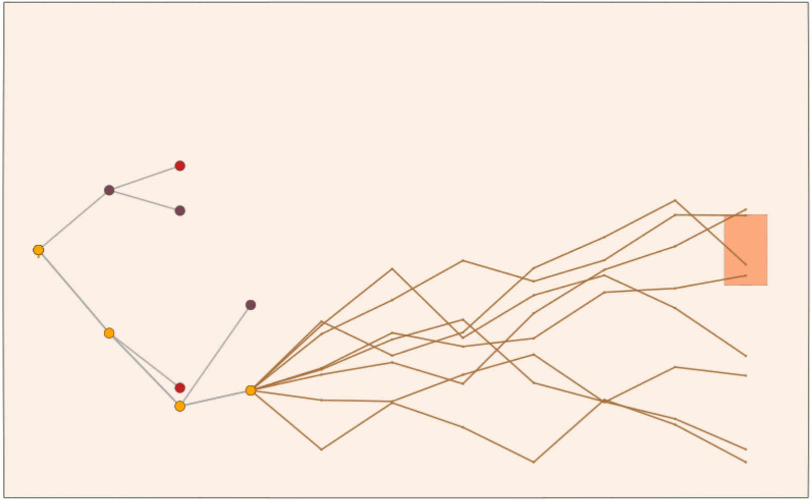
- **Selection**: Walk down tree to leaf z_{t-1}
- **Expand**: Sample ~ 5 next steps z_t, pick one at random
- **Rollouts**: Sample steps $z_{t+1} \dots z_T$
- **Backprop**: Update nodes counts $N(z_{1:i})$ and wins $w(z_{1:i})$ for parents

Generalization: MCTS

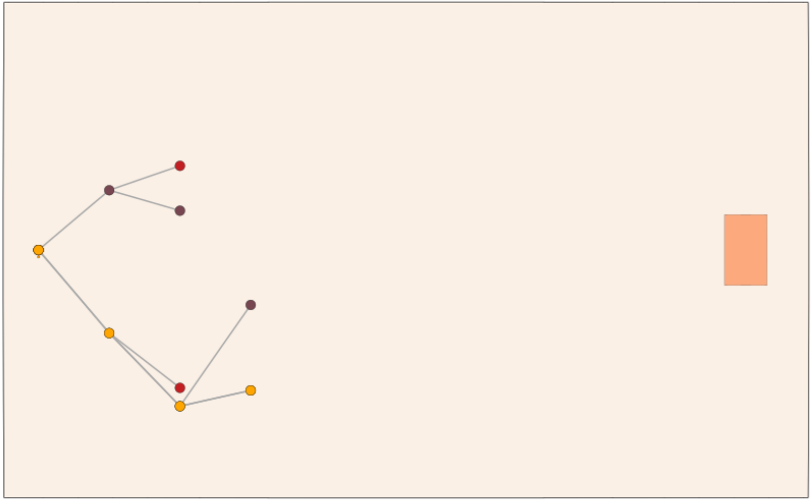
Generalization: MCTS



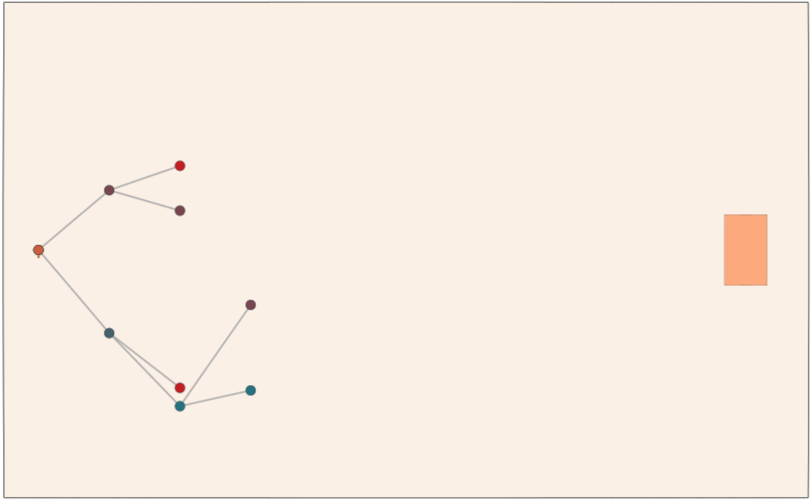
Generalization: MCTS



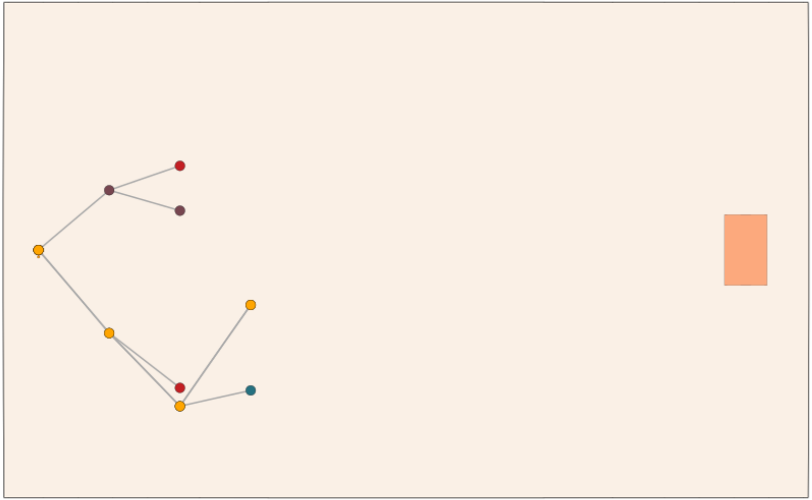
Generalization: MCTS



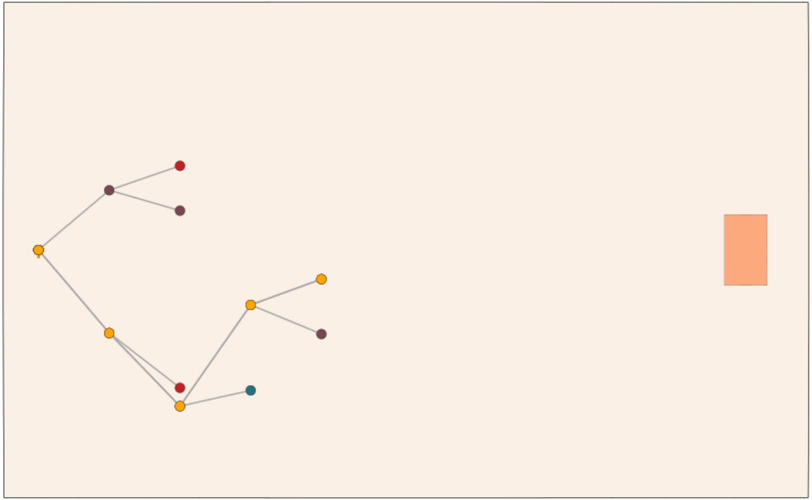
Generalization: MCTS



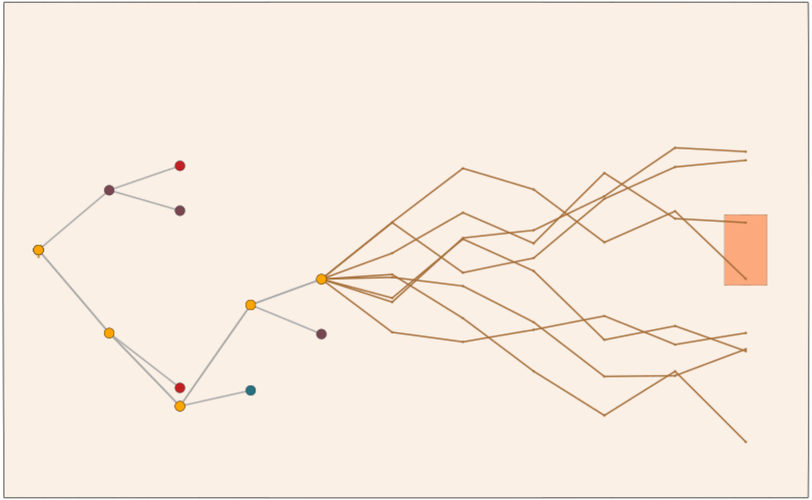
Generalization: MCTS



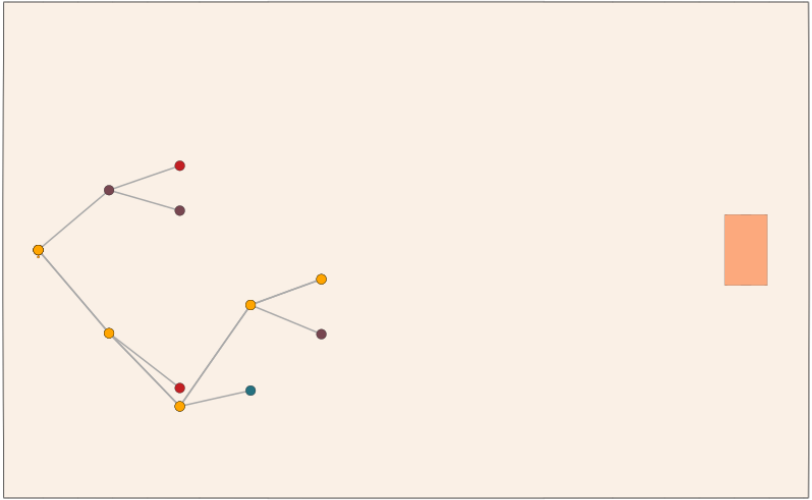
Generalization: MCTS



Generalization: MCTS



Generalization: MCTS



Exploration

[Kocsis and Szepesvári, 2006]

- MCTS-UCB explores states based on wins and amount of explorations

$$\frac{w(z_{1:t})}{N(z_{1:t})} + \alpha\sqrt{\frac{\ln N(z_{1:t-1})}{N(z_{1:t})}}$$

- Less strict search process

Learning from Search

[Xie et al., 2024, Putta et al., 2024]

- MCTS tree provides path preferences
- Can be used for preference learning (e.g. DPO)
- Alternative to learning on chains

Is this o1?

- ✓ Major demonstrated RL result
- ✓ Scales to more train-time search

Is this o1?

- ✓ Major demonstrated RL result
- ✓ Scales to more train-time search

- × Costly to maintain open states
- × More complex algorithmically
- × OpenAI comments / rumors

The Suspects

- Guess + Check
- Process Rewards
- Search / AlphaZero
- Learning to Correct

What does exploration look like?

- Game Playing - Explore alternative moves.
- Language - Nearly infinite ”moves”
- Exploration to learn strategies

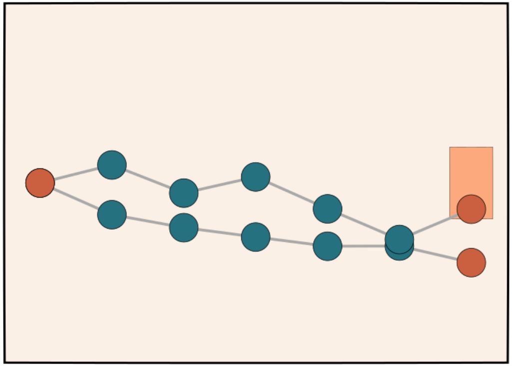
Suspect 4: Learning to Correct

- 1) Start with failed CoT
- 2) Search to find successful corrections
- 3) Train on full CoT

Framework: Self-Correction

[Welleck et al., 2022]

- Aim: Find similar CoT pairs z', z'' where z'' is better.
- Train the model to improve upon z'



[Gandhi et al., 2024]

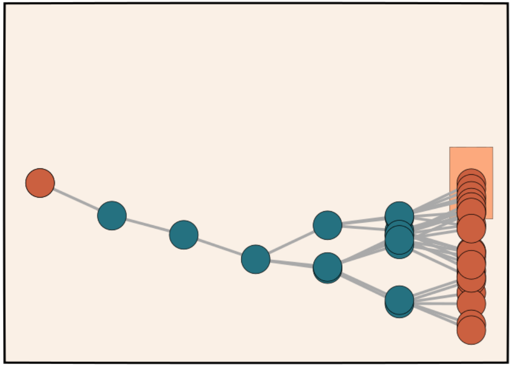
Challenges: Learning Correction

- Collapse: Model may learn to just ignore negative
- Distribution Shift: Actual mistakes may deviate from examples

[Gandhi et al., 2024]

RL from Mistakes

- Start with z'
- Learn to correct from verifier



Empirical Results

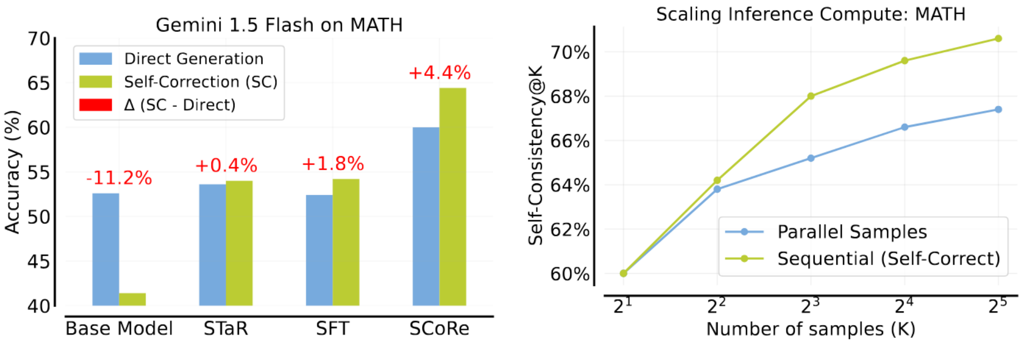


Figure 1 | **Left:** ***SCoRe*** achieves state-of-the-art self-correction performance on MATH; **Right:** ***SCoRe*** inference-time scaling: spending samples on *sequential* self-correction becomes more effective than only on *parallel* direct samples (Section 6.2).

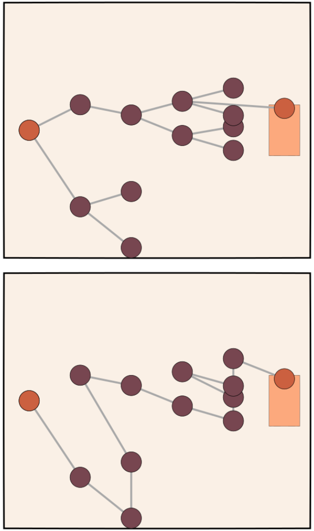
Generalization: Stream of Search

[Gandhi et al., 2024]

- Find $z^*_{1:T}$ as optimal length CoT

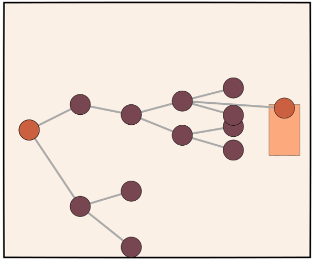
- Find $z'_{1:T'}$ with $T' > T$ through backtracking tree search

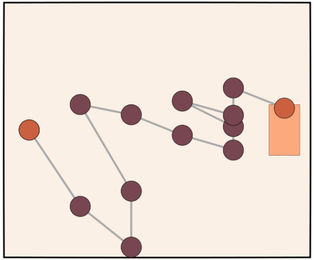
- Train model on $z'_{1:T'}$



From Tree to Stream

- Tree search explores multiple paths
- Stream presents a linear sequence
- Allows model to mistakes in stream





Is this o1?

- ✓ Learns to correct and plan
- ✓ Single test-time model

Is this o1?

- ✓ Learns to correct and plan
- ✓ Single test-time model
- × Complex training process
- × Limited empirical evidence

Outline

Introduction

The Clues

Technical Background

The Suspects

Implications

Replication

- Critical to have open-source versions
- Systems aspects are different
- Versions may not look the same

Research Implication

- Beyond Emergent Ablities

Research Implication

- Beyond Emergent Ablities
- Inference Time Systems

Research Implication

- Beyond Emergent Ablities
- Inference Time Systems
- From Prompting to Specification

Research Implication

- Beyond Emergent Ablities
- Inference Time Systems
- From Prompting to Specification
- Evaluations Need to Be Much Harder

Research Implication

- Beyond Emergent Ablities
- Inference Time Systems
- From Prompting to Specification
- Evaluations Need to Be Much Harder
- CoT Change Interpretability

Thank You

`https://github.com/srush/awesome-o1`

Reference I

[Ankner et al., 2024] Ankner, Z., Paul, M., Cui, B., Chang, J. D., and Ammanabrolu, P. (2024).

Critique-out-loud reward models.

arXiv [cs.LG].

[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 and Sandholm, 2017] Brown, N. and Sandholm, T. (2017).

Libratus: The superhuman AI for no-limit poker.

In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, California. International Joint Conferences on Artificial Intelligence Organization.

Reference II

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020).

Language models are few-shot learners.

arXiv [cs.CL].

[Cobbe et al., 2021] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021).

Training verifiers to solve math word problems.

arXiv [cs.LG].

Reference III

[Feng et al., 2023] Feng, X., Wan, Z., Wen, M., McAleer, S. M., Wen, Y., Zhang, W., and Wang, J. (2023).
Alphazero-like tree-search can guide large language model decoding and training.
arXiv [cs.LG].

[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 IV

[Hendrycks et al., 2021a] Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D., and Steinhardt, J. (2021a).
Measuring coding challenge competence with APPS.
arXiv [cs.SE].

[Hendrycks et al., 2021b] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021b).
Measuring massive multitask language understanding.

[Hendrycks et al., 2021c] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021c).
Measuring mathematical problem solving with the MATH dataset.
arXiv [cs.LG].

Reference V

[Hosseini et al., 2024] Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. (2024).

V-STar: Training verifiers for self-taught reasoners.

In *First Conference on Language Modeling*.

[Hubert et al., 2021] Hubert, T., Schrittwieser, J., Antonoglou, I., Barekatain, M., Schmitt, S., and Silver, D. (2021).

Learning and planning in complex action spaces.

arXiv [cs.LG].

[Jones, 2021] Jones, A. L. (2021).

Scaling scaling laws with board games.

arXiv [cs.LG].

Reference VI

[Kazemnejad et al., 2024] Kazemnejad, A., Aghajohari, M., Portelance, E., Sordoni, A., Reddy, S., Courville, A., and Roux, N. L. (2024).
VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment.
arXiv [cs.LG].

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006).
Bandit based monte-carlo planning.
In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 282–293. Springer Berlin Heidelberg, Berlin, Heidelberg.

[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].

Reference VII

[Nakano et al., 2021] Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., Jiang, X., Cobbe, K., Eloundou, T., Krueger, G., Button, K., Knight, M., Chess, B., and Schulman, J. (2021).

WebGPT: Browser-assisted question-answering with human feedback.

arXiv [cs.CL].

[Neal and Hinton, 1998] Neal, R. M. and Hinton, G. E. (1998).

A view of the em algorithm that justifies incremental, sparse, and other variants.

In *Learning in Graphical Models*, pages 355–368. Springer Netherlands, Dordrecht.

[Nye et al., 2021] Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. (2021).

Show your work: Scratchpads for intermediate computation with language models.

arXiv [cs.LG].

Reference VIII

[OpenAI, 2024] OpenAI (2024).
Learning to reason with LLMs.
`https://openai.com/index/learning-to-reason-with-llms/`.
Accessed: 2024-10-29.

[Putta et al., 2024] Putta, P., Mills, E., Garg, N., Motwani, S., Finn, C., Garg, D., and Rafailov, R. (2024).
Agent Q: Advanced reasoning and learning for autonomous AI agents.
arXiv [cs.AI].

[Silver et al., 2017] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017).
Mastering chess and shogi by self-play with a general reinforcement learning algorithm.
arXiv [cs.AI].

Reference IX

[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.
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].

[Sutton, 2019] Sutton, R. (2019).
The bitter lesson.

Reference X

[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.
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].

[Wang et al., 2022] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2022).
Self-consistency improves chain of thought reasoning in language models.
arXiv [cs.CL].

Reference XI

[Wei et al., 2022] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2022).

Chain-of-thought prompting elicits reasoning in large language models.

arXiv [cs.CL], pages 24824–24837.

[Welleck et al., 2024] Welleck, S., Bertsch, A., Finlayson, M., Schoelkopf, H., Xie, A., Neubig, G., Kulikov, I., and Harchaoui, Z. (2024).

From decoding to meta-generation: Inference-time algorithms for large language models.

arXiv [cs.CL].

[Welleck et al., 2022] Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. (2022).

Generating sequences by learning to self-correct.

arXiv [cs.CL].

Reference XII

[Xie et al., 2024] Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. (2024).

Monte carlo tree search boosts reasoning via iterative preference learning.

arXiv [cs.AI].

[Yarowsky, 1995] Yarowsky, D. (1995).

Unsupervised word sense disambiguation rivaling supervised methods.

In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* -, Morristown, NJ, USA. Association for Computational Linguistics.

[Zelikman et al., 2022] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. (2022).

STaR: Bootstrapping reasoning with reasoning.

arXiv [cs.LG].

Reference XIII

[Zhang et al., 2024] Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. (2024).

Generative verifiers: Reward modeling as next-token prediction.

arXiv [cs.LG].